НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

УДК 007.3

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Клієнт-серверна аналітична система управління робочим часом*»

Виконав: студент *4* курсу, групи *ІС-52*

Косяк Олександр Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. вик. Проскура С. Л.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст. вик. Халус О. А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Кисленко Ю.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)

“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Косяку Олександру Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема проекту « *Клієнт-серверна аналітична система управління робочим часом*»

керівник проекту *Проскура Світлана Леонідівна, ст.вик.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*04*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використання

2. Схема бази даних

3. Схема структурна класів

4. Схема структурна послідовності

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«15» лютого 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*15.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*27.03.2019*	
3.	*Постановка та формалізація задачі*	*29.03.2019*	
4.	*Розробка інформаційного забезпечення*	*08.04.2019*	
5.	*Алгоритмізація задачі*	*17.04.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*21.04.2019*	
7.	*Розробка програмного забезпечення*	*25.05.2019*	
8.	*Налагодження програми*	*27.05.2019*	
9.	*Виконання графічних документів*	*28.05.2019*	
10.	*Оформлення пояснювальної записки*	*29.05.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ О. М. Косяк
(підпис)

Керівник проекту _____ С. Л. Проскура
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5213.1181-с.ПЗ	Пояснювальна записка	53	
2	А4	ДП ІС-5213.1181-с.ТЗ	Технічне завдання	9	
3	А4	ДП ІС-5213.1181-с.ПМВ	Програма та методика випробувань	0	
			Матеріали графічні		
4	А3	ДП ІС-5213.1181-с.ССП	Схема структурна послідовності	2	
6	А3	ДП ІС-5213.1181-с.СБД	Схема бази даних	1	
7	А3	ДП ІС-5213.1181-с.ССК	Схема структурна класів програмного	2	
			забезпечення		
8	А3	ДП ІС-5213.1181-с.ССВВ	Схема структурна варіантів використання	1	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5213.1181-с.ВДП		
Розробн.		Косяк О.М.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Проскура С.Л.					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н/контр.		Халус О. А.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Клієнт-серверна аналітична система управління робочим часом

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 20 рисунків, 10 таблиць, 1 додаток, 7 джерел.

Дипломний проект присвячений розробці комплексу задач клієнт-сервеної аналітичної програми, з управління робочим часом.

Описані вхідні та вихідні дані програми, структура бази даних.

Обгрунтовано й пояснено вибір методів вирішення задач, та наведено їх пояснення.

Описана структура програми, що була створена.

Описане керівництво користувача, та проведено випробування програми.

Ключові слова: КЛІЄНТ-СЕРВЕРНА, АНАЛІТИЧНА, УПРАВЛІННЯ РОБОЧИМ ЧАСОМ, ГЕНЕРАЦІЯ РОЗКЛАДІВ

ABSTRACT

Structure and scope of work. Explanatory note of the diploma project consists of six sections, contains 20 drawings, 10 tables, 1 applications, 7 sources.

The diploma project is devoted to the development of a complex of tasks of the client-servant analytical program, on working time management.

Describes the input and output data of the program, the structure of the database.

The choice of methods of solving problems is explained and explained, and their explanation is given.

The structure of the program that was created is described.

The user manual is described, and the program has been tested.

Key words: CLIENT-SERVER, ANALYTICAL, WORKING-TIME MANAGEMENT, SCHEDULES GENERATION

ЗМІСТ

ВСТУП

Дипломний проект присвячено розробці клієнт-серверної програми що відповідає за автоматизацію процесу створення розкладу користувачів. Нерідко трапляються випадки коли потрібно швидко розпланувати певний набір задач. Не дивлячись на тривіальність даного процесу, рівень його автоматизації в представлених на ринку аналогах зводиться до звичайної заміні паперових записок на електронний варіант.

На відміну від наявних на ринку аналогів, так званих «органайзерів», програмний продукт що було розроблено в ході дипломної роботи автоматично генерує розклад на основі набору задач що вводить користувач.

Практичне значення одержаних результатів. Було розроблено алгоритм що генерує розклад, на основі заданих користувачем пріоритетів, та часу до «дедлайну» конкретної задачі.

Публікації. Результати роботи були опубліковані у вигляді тез доповіді у науковому збірнику «Секція кафедри автоматизованих систем обробки інформації і управління. Матеріали конференції» ІІ Всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління (с. 117-121).»

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Зараз на ринку представлена велика кількість програм, так званих «Планувальників», які відрізняються зазвичай лише дизайном, а функціонал у них в більшості однаковий: користувач сам, вручну планує свої дії. При плануванні свого робочого часу людина може спиратись лише на себе і свій досвід. В нашу задачу входить: допомогти користувачу розпланувати свій час так, щоб він максимально ефективно виконував свою роботу (на основі різних даних будуть надані автоматичні рекомендації для поліпшення існуючого розпорядку, або згенерований оптимальний розпорядок «з нуля», на основі даних введених користувачем), при цьому мінімізувати шкідливий вплив на його організм (через постійне сидіння в одному положенні, перевтому, порушення режиму харчування) що відповідно знизить рівень уваги та ефективність праці.

Люди які розплановують свій час часто користуються різними методами та способами цього планування, вони використовують різноманітні програми та застосування, починаючи з папірців на холодильнику і завершуючи різноманітними текстовими редакторами та різними щоденниками. Але суть процесу планування завжди одна, людина сама розміщує свої задачі в певному порядку, інколи розкидуючи їх по різним дням. Зазвичай, для такого планування витрачається багато часу, адже кожну задачу потрібно вносити вручну, наперед слід визначити усі варіанти взаємоположення задач в часі та вибрати оптимальне, також не виключені ситуації коли про якусь задачу можна просто забути. Саме тому дане планування робочого часу користувачем не є ефективним.

Так для планування свого робочого часу, скажімо, на 2 тижні потрібно витратити близько години. Також при такому плануванні рідко враховуються

такі фактори як: час на перерви, протяжність цих перерв, час на те щоб «переналаштувати» свій розум на нову задачу, адже це може викликати потреби «вникати» у проблему заново. Саме тому набагато ефективніше розпланувати час так, щоб виконання 1 задачі завершувалося або на перерву, апо по завершенню робочого часу.

Також слід враховувати усі «дедлайни» для кожної з задач, що в свою чергу теж потребує час на осмислення. Крім того потрібно весь час тримати в голові пріоритетність кожної задачі.

Не слід забувати про те ,що якщо людина працює на себе і є так званим «фрилансером», або має незвичний графік, наприклад 3 робочі години зранку та 5 в вечері то розпланувати задачі для неї стає ще складнішим завданням, і не рідко що саме в таких ситуаціях виникають потреби працювати понаднормово, що може погано сказатись на здоров'ї людини.

Саме при плануванні гнучких графіків виникають ситуації коли потрібно максимально ефективно розпланувати свій час, враховуючи усі можливі ситуації що можуть завадити виконанню роботи. Набагато краще коли в щоденнику стоїть не просто потрібно зробити ще 5 проектів за 15 днів, а розписано що конкретно потрібно робити щодня, для виключення ситуації коли потрібно зробити всі ті самі 5 проектів, але вже за 2 дні. Адже є багато досліджень на основі яких можна складати зручні розклади робочого часу.

1.1.1Опис процесу діяльності

Зазвичай для створення розкладу користувачем(що являється результатом процесу планування використання робочого часу) використовують звичайні паперові записники, щоденники та плани. На даний момент процес автоматизації створення розкладу не сильно просунувся вперед. В основному зараз як і раніше вся робота з планування повністю

виконується людиною, а вся допомого комп'ютера зводиться до заміни паперових аналогів на електронний записник. Навіть найбільш просунуті «органайзери», в своїй суті є звичайними записниками і в своєму функціоналі не сильно відрізняються від звичайного текстового редактору.

Саме тому було вирішено створити програму що зможе максимально автоматизувати процес створення розпорядку робочого часу. Нижче ми можемо побачити схему структурну діяльності (рисунок 1.1).

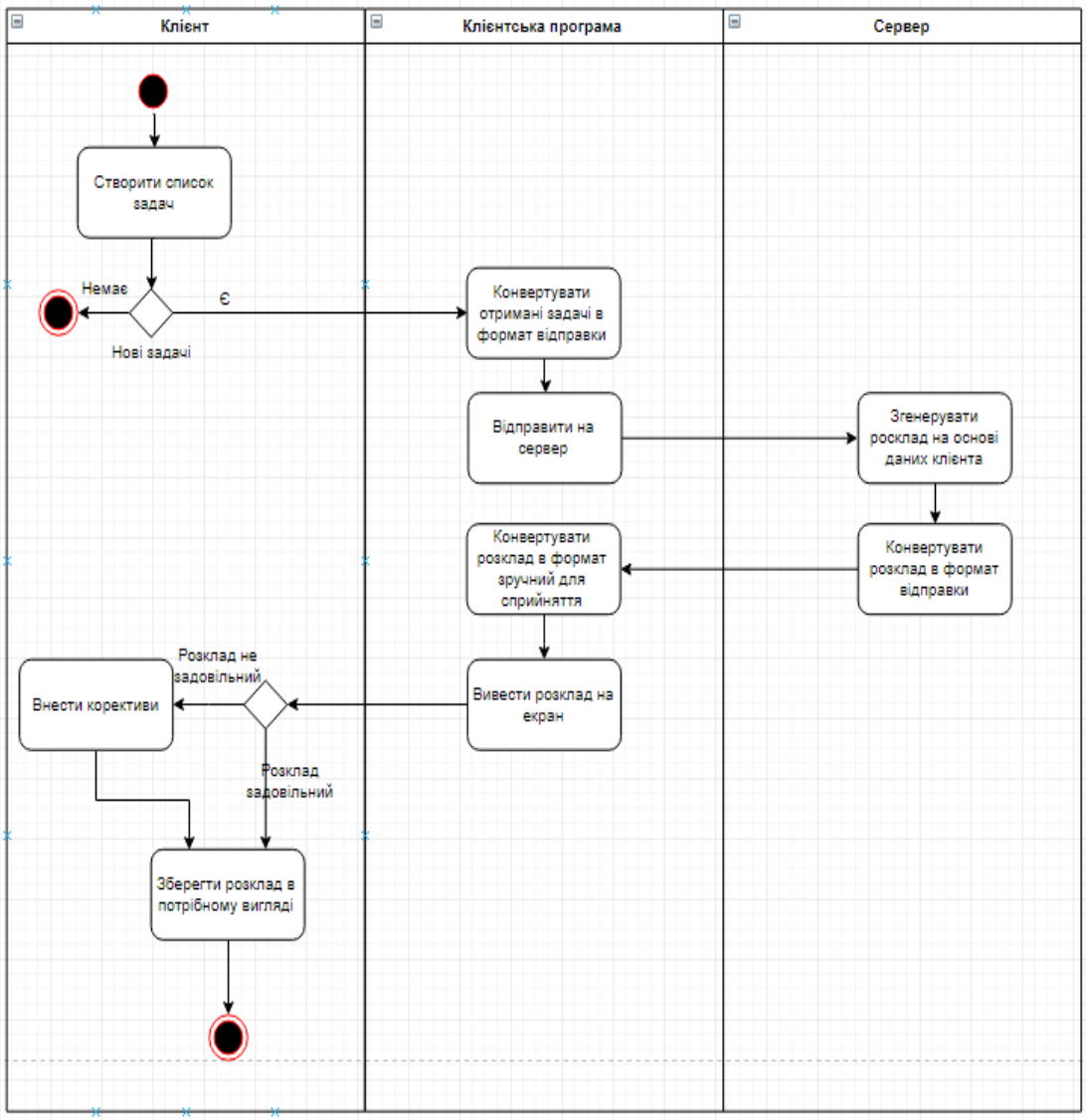


Рисунок 1.1 – Схема структурна діяльності

Отже ми можемо побачити що наша система отримує від користувача не структурований набір задач, та на його основі генерує розклад. Даний процес відбувається в кілька етапів:

- отримані від клієнта відомості про задачу оброблює клієнтський додаток;
- після попередньої обробки введеної інформації відбувається її відправка на серверний додаток (або простіше кажучи на сервер);
- відбувається структурування внесеної користувачем інформації про задачі, та на основі пріоритетів користувача, й дат на створення задачі відбувається генерування розкладу та його відправка користувачу.
- далі користувач може внести корективи в розклад, та зберегти остаточний варіант розкладу.

Таким чином час що буде затрачено на планування буде знижено щонайменше в половину, та значно спростить сам процес планування, знявши частину аналітичної роботи з користувача.

Для розробки програмного продукту було вирішено використовувати клієнт-серверну архітектуру, оскільки це дозволить вносити корективи в систему генерації розкладів не завдаючи клієнтам зайвих труднощів. Також дана архітектура дозволить ефективно збирати статистичні дані з використання програми, на основі яких можна буде ефективно та оперативно вносити корективи в роботу системи. Не менш важливим фактором є те, що клієнт-серверна архітектура дозволить максимально знизити навантаження на систему користувача, та перекласти усі ці обов'язки на сторону сервера. Також дана архітектура дозволить користувачу отримувати доступ до своїх розкладів з будь-якого комп'ютера, просто зайшовши до системи під своїм унікальним ім'ям користувача.

1.1.2Опис функціональної моделі

Опис функцій, що будуть використовувати користувачі буде наведено у вигляді діаграми IDEF0 (рисунок 1.2 - рисунок 1.5).

Методологія IDEF0 передбачає побудову ієрархічної системи діаграм – одиничних описів фрагментів системи. Спочатку проводиться опис системи в цілому і її взаємодія з оточуючим світом (рисунок 1.2), після чого проводиться функціональна декомпозиція – система розбивається на підсистеми і кожна підсистема описується окремо (рисунок 1.3 – рисунок 1.5). Після чого кожна з підсистем в свою чергу може бути розбита на менші підсистеми, і так далі, доки не буде досягнуто потрібного рівня деталізації [1].

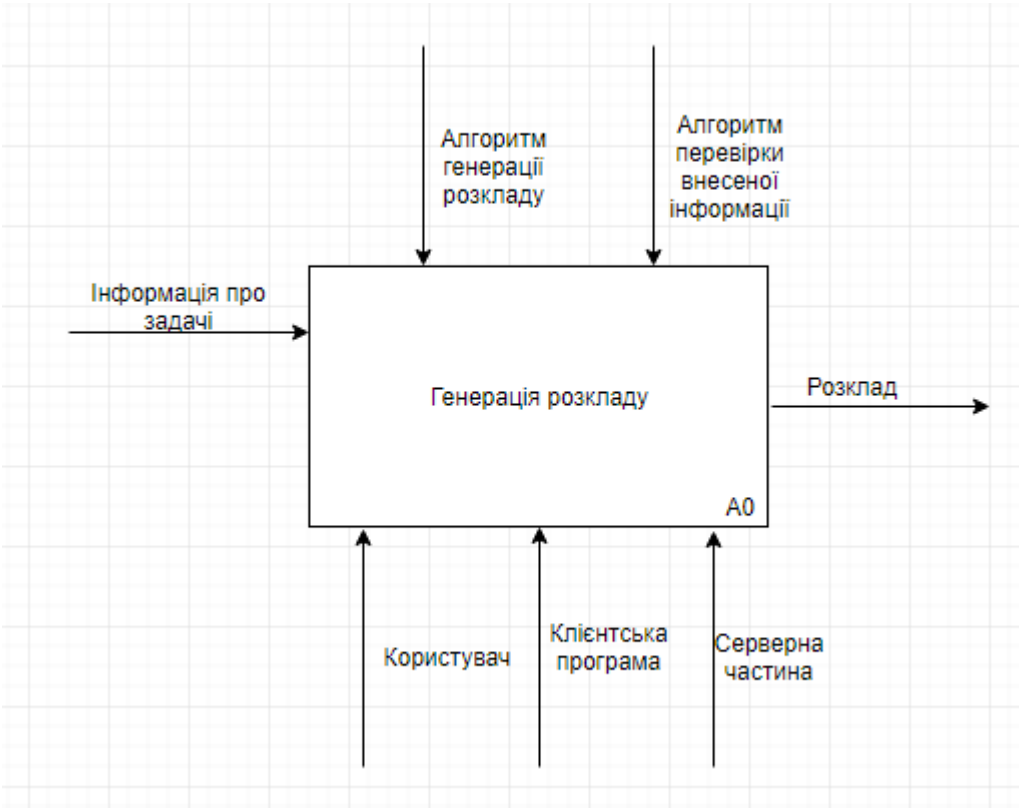


Рисунок 1.2 – Схема структурна контекстної моделі системи

На діаграмі відображено лише один блок – головна функція нашої системи, а саме генерація розкладок користувача, на основі внесеної користувачем задачі.

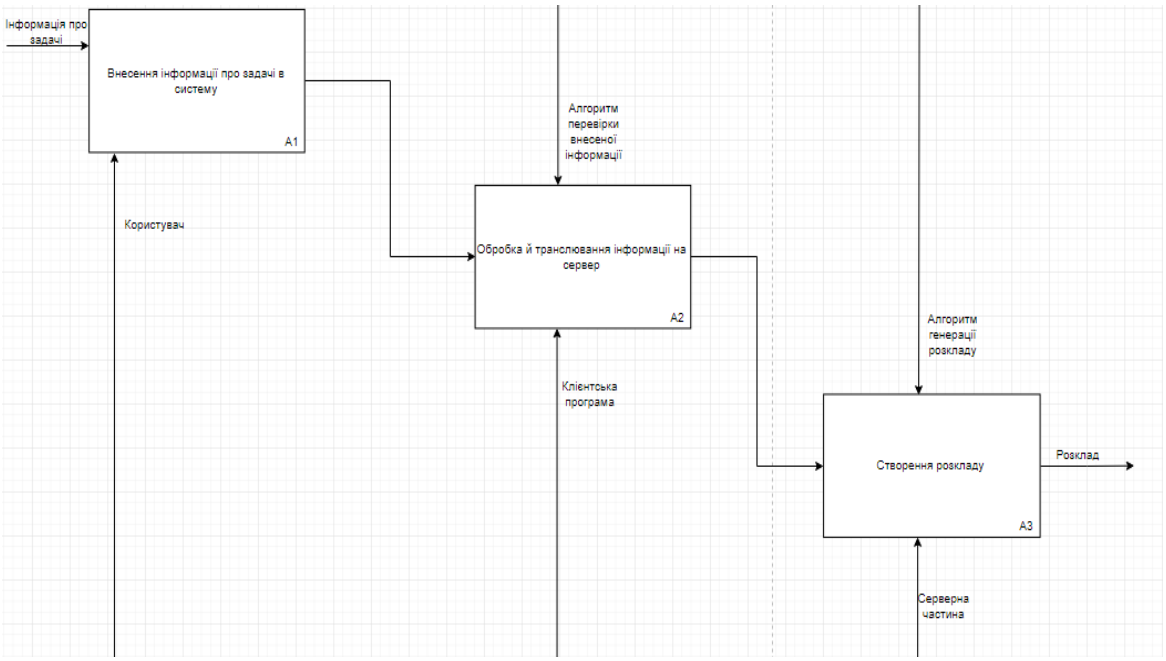


Рисунок 1.3 – Схема структурна функціональної моделі IDEF0

На рисунку 1.3 можна побачити перший рівень декомпозиції системи, наша головна функція (генерація розкладу) розбивається на три задачі: внесення інформації, попередня обробка інформації, та складання розкладу.

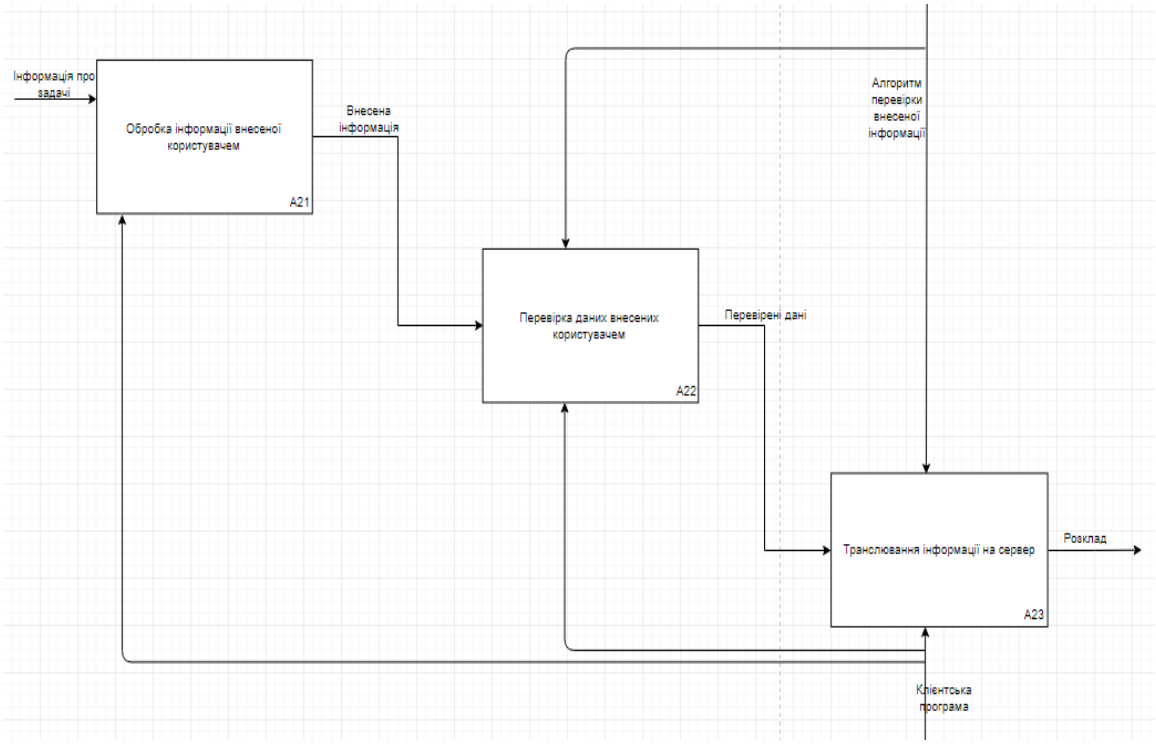


Рисунок 1.4 – Схема структурна функціональної моделі IDEF0

На рисунку 1.4 відображено декомпозицію функції що відповідає за попередню обробку інформації на клієнтському додатку. Відображено три підзадачі: обробка інформації внесеної користувачем, перевірка внесеної користувачем інформації, та відправка інформації на сервер.

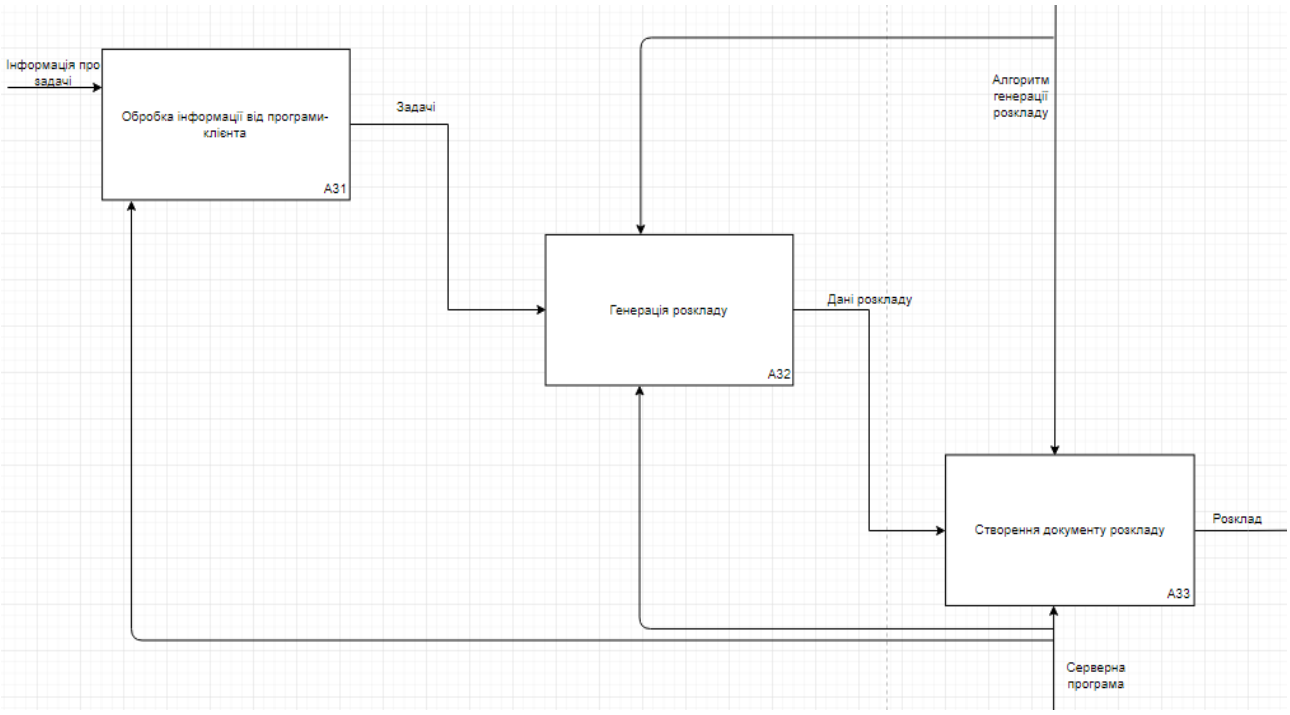


Рисунок 1.5 – Схема структурна функціональної моделі IDEF0

На рисунку 1.5 відображена декомпозиція функції створення розкладу на сервері. Основна задача розбивається на три підзадачі: обробка отриманої від клієнтського додатку інформації, генерування розкладу за допомогою алгоритму створення розкладу, та генерація файлу розкладу.

Відповідно до заявленої вище функціональної моделі, було сформу-льовано перелік функціональних вимог до системи.

Таблиця 1.1 – Функціональні вимоги

Функціональні вимоги	Пріоритет
1. Система надає можливість користувачу ввести інформацію про задачі	Високий
1.1. Система надає можливість переглянути введені задачі	Високий

Продовження таблиці 1.1

Функціональні вимоги	Пріоритет
2. Система надає можливість користувачу редагувати розклад	Високий
3. Система надає можливість переглянути меню	Високий
3.1. Система надає можливість переглянути розклади створені раніше	Високий
3.2. Система надає можливість переглянути інформацію про програму	Високий
3.3. Система надає можливість переглянути календар	Високий
3.4. Система надає можливість робити записи в блокнот	Високий
3.5. Система надає можливість змінювати тему (фон) додатку	Середній
3.6. Система надає можливість переглянути інформацію про свої функції	Високий
3.7. Система надає можливість переглянути інформацію про авторів	Середній

Нефункціональні вимоги до системи:

- система має за короткий час видати відповідь на запит користувача;
- система повинна мати інтуїтивно зрозумілий інтерфейс.

1.2 Огляд наявних аналогів

На сьогодні вже розроблено ряд програмного забезпечення, яке допомагає людині розробляти розклад, та полегшувати планування робочого часу.

Наприклад існує програма YouTrack — онлайн баг-трекер і інструмент для управління задачами, проектами і Agile-процесами. Локалізований на

російській мові. Безкоштовна версія для 10 користувачів. Розроблена JetBrains [2].

Рисунок 1.6 – Веб-застосування YouTrack

Основна ідея даного застосування полягає в тому що це програма для розробників, що дозволяє ефективно контролювати розробку програм, ефективно управляти проектом та перевіряти стан робіт на проекті.

Переваги даного програмного забезпечення в тому, що воно надає дуже потужний інструментарій по управлінню проектами, та надзвичайно велику кількість різноманітних ефективних особливостей що дозволяє корегувати й контролювати роботу над великими проектами, та в свою чергу підвищує ефективність управління. Слід зауважити що дана програма також дозволяє спланувати роботу в тому числі й цілого колективу, над певною задачею, що в свою чергу є вагомою перевагою.

Недоліки полягають частково в його перевагах, для одного користувача весь програмний функціонал є надмірним, а для людей, що займаються діяльністю, що не сильно зв’язана з розробкою програмного забезпечення він майже не придатний.

Також він не сильно скорочує час що витрачається на планування робочого розпорядку, оскільки грубо кажучи, процес планування не сильно

буде відрізнятись, від написання задач на папірцях та їх прикріплення на стіну (не враховуючи додатковий функціонал що спрощує та прискорює розробку програмних продуктів).

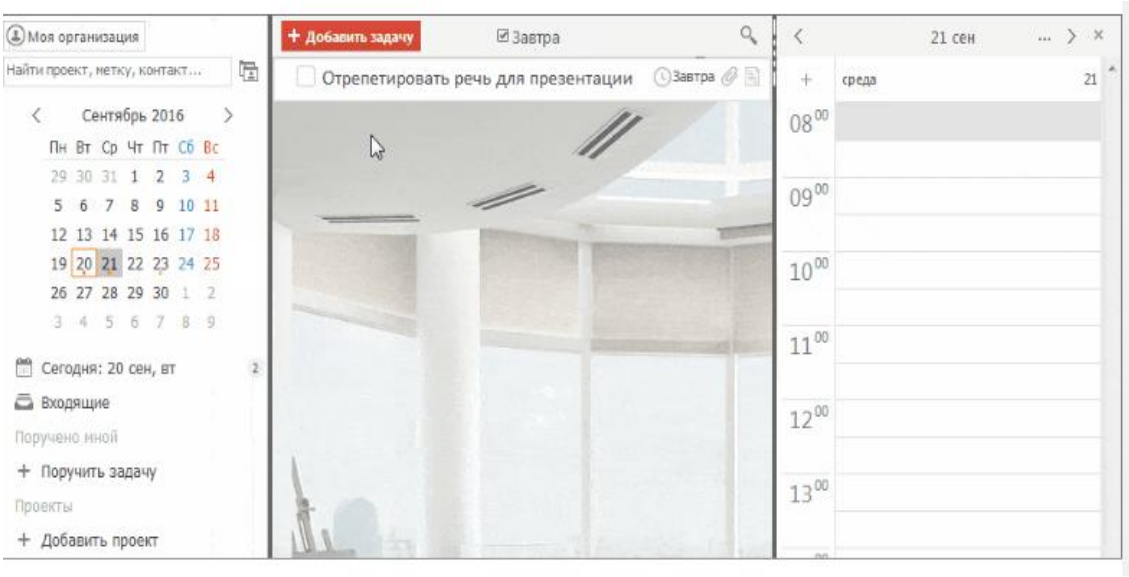


Рисунок 1.7 – Програма ЛидерТаск

Програма ЛидерТаск – десктопний органайзер розроблений «Органайзер ЛидерТаск» [3].

Перевагаю даного застосування є те що воно розроблене як планувальник задач, отже розраховане на значно більшу аудиторію ніж YouTrack, що в свою чергу означає що розібратись в інтерфейсі й функціоналі даного додатку буде значно простіше.

Недоліки даного додатку в тому, що по суті він не вирішує задачі скорочення часу планування, користувач сам розписує задачі на кожен день, сам визначає їх порядок, програма ніяк не допомагає в вирішення конфліктів при накладанні кінцевих строків виконання задачі.

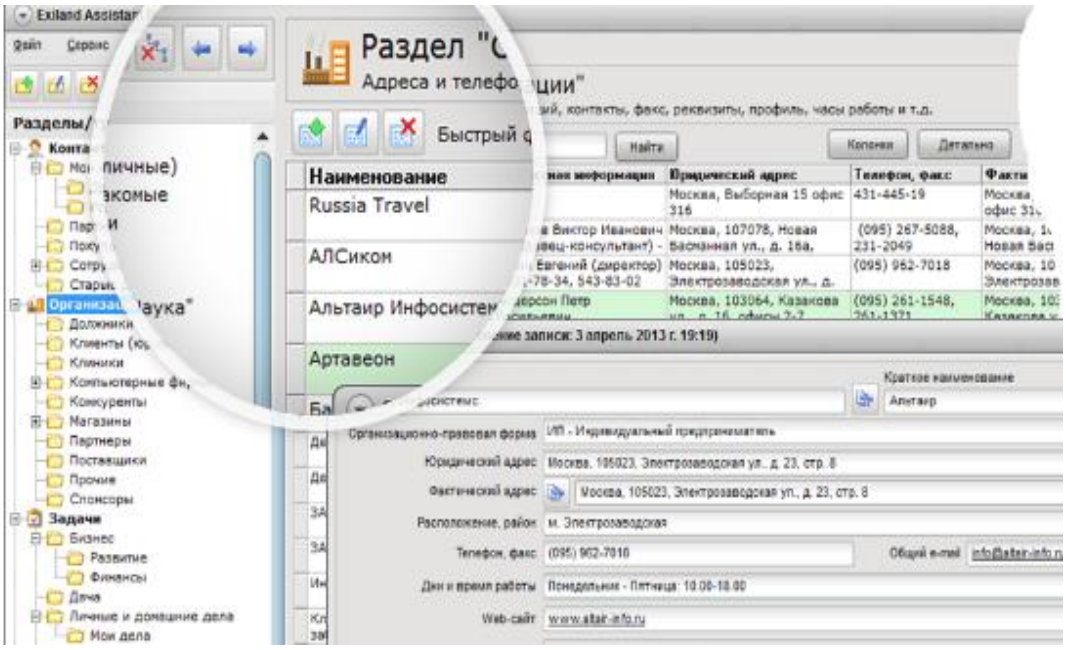


Рисунок 1.8 – Програма Exiland Assistant Free

Exiland Assistant Free – це програма, головною задачею якої є ведення бази данних задач, друзів, зберігати різноманітну інформацію, тощо [4].

Перевагою є те, що в програми надзвичайно велика кількість можливостей для збереження різноманітних деталей про задачу, людину, подію.

Недоліком є те, що інтерфейс програми дуже перевантажений, та при цьому оформлений в стилі старих файл-менеджерів, що може відлякнути користувачів. Також час що потрібен для заповнення всієї інформації в програму просто неймовірний.

Отже як бачимо, незважаючи на величезну кількість програм-органайзерів на ринку, весь іх функціонал не сильно відрізняється від звичайного паперового щоденника, і рівень автоматизації в них приблизно зведений до перегортання сторінок натисканням клавіші.

1.3 Постановка задачі

Мета роботи – підвищення ефективність використання робочого часу користувача при зменшенні часу затраченого на планування розпорядку.

Для досягнення поставленої мети необхідно виконати наступні задачі:

-знайти та проаналізувати аналоги на предмет ефективності роботи;

-вирішити задачу автоматизації процесу генерації розпорядку;

-зібрати інформацію про ефективність різних підходів до використання робочого часу;

-створити ПЗ, що дозволить багатьом користувачам використовувати потужності серверу, а не навантажувати власний комп’ютер;

1.3.1Призначення розробки

Призначенням розробки є підтримка процесу планування робочого часу користувачем за рахунок генерації розкладу.

1.3.2Цілі та задачі розробки

Цілями розробки є:

- прискорення процесу автоматизації створення розкладу;
- генерація процесу планування робочого часу користувача;
- зменшення часу витраченого на процес планування.

Для досягнення цілей, повинні бути вирішені наступні задачі:

- створити систему авторизації;
- надати можливість редагувати згенерований розклад;
- реалізувати збір інформації про задачі.

Висновок до розділу

В даному розділі, був проведений опис предметного середовища, було визначено вимоги, було знайдено аналоги розроблюваного програмного продукту та зроблено порівняльну характеристику, а також було визначено призначення розробки, цілі та задачі що потрібно вирішити для досягнення поставлених цілей. Виходячи з висновків порівняння з аналогами можемо

зробити висновок, що наше застосування буде корисним, та ефективнішим за аналоги.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідними даними системи є данні що надходять від користувача. Їх можна розділити на дві частини. Перша, загальна інформація про розклад що потрібно згенерувати.

- дата початку планування;
- дата завершення планування;
- час робочого дня;
- вихідні дні (якщо вони є).

Друга частина, це інформація про конкретну задачу, в кожному розкладі відповідно може бути довільна кількість задач, для кожної з яких слід вказати:

- дату початку роботи над задачею;
- дату завершення роботи над задачею;
- назву задачі;
- пріоритет задачі;
- чи може задача перериватись;
- час на виконання задачі;
- опис задачі.

Також є додатковий функціонал, а саме «блокнот для записів», який отримує на вхід наступну інформацію:

- дату;
- назву задачі;
- опис.

2.2 Вихідні дані

Вихідними даними даного програмного продукту є згенерований HTML файл. Даний файл можливо переглянути у будь-якому браузері на будь-якому пристрої.

2.3 Опис структури бази даних

Для функціонування клієнт-серверної аналітичної програми з управління робочим часом було створено базу даних що складається з трьох таблиць:

- реєстрація;
- розклад користувача;
- задачі.

В склад таблиці «Реєстрація» входять поля:

- логін;
- пароль.

Опис таблиці «Реєстрація» наведено у таблиці 2.1.

Таблиця 2.1 – «Реєстрація»

Код	**Опис**	**Тип даних**	**Обов'язкове**	**Унікальне**	**Первинний ключ**	**Зовнішній ключ**
Id	Ідентифікатор питання	integer	X	X	X	
UserLogin	логін користувача	Nvarchar[50]	X			
UserPass	Пароль користувача	Nvarchar[50]	X			

Для розміщення та збереження інформації про розклад користувача використано таблицю «Розклад користувача» в яку входять поля:

- ідентифікатор користувача;
- час генерації;
- дані розкладу;
- вихідні дні;
- час робочого дня;
- час початку планування;
- час завершення планування.

Опис та типи містяться в таблиці 2.2.

Таблиця 2.2 – «Розклад користувача»

Код	**Опис**	**Тип даних**	**Обов'язкове**	**Унікальне**	**Первинний**	**Зовнішній**
Id	Ідентифікатор розкладу	integer	X	X	X	
UserTTs	Дані розкладу	Nvarchar[MAX]	X			
userID	Ідентифікатор користувача	integer	X			
DateOfCreate	Дата створення	date	X			
DateStartTT		date	X			
DateEndTT		date	X			
WorkDay		integer	X			
Weekends		Nvarchar[10]	X			

Тертя таблиця, яка взаємопов'язана з вищеописаними таблицями містить наступні поля:

- назва задачі;
- опис задачі;
- дата початку задачі;
- дата кінця задачі;
- пріоритет;
- час на виконання задачі;
- ідентифікатор розкладу.

Ці поля призначені безпосередньо для збереження задач.

Таблиця 2.3 – «Задачі»

Код	**Опис**	**Тип даних**	**Обов'язкове**	**Унікальне**	**Первинний**	**Зовнішній**
Id	Ідентифікатор зображення	integer	X	X	X	
TaskName	назва задачі	Nvarchar[50]	X			
TaskDeskription	опис задачі	Nvarchar[MAX]	X			
TaskStartDate	дата початку задачі	date	X			
TaskEndDate	дата кінця задачі	date	X			
TaskUserPriority	пріоритет	Nvarchar[10]	X			
taskLenth	час на виконання задачі	integer	X			

Продовження таблиці 2.3

Код	Опис	Тип даних	Обов'язкове	Унікальне	Первинний ключ	Зовнішній ключ
TTid	ідентифікатор розкладу	integer	X			

Висновок до розділу

У даному розділі наведені вхідні та вихідні дані аналітичної системи управління робочим часом користувача.

Надано описи структури таблиць бази даних та детальну інформацію по кожній сутності з описом полів. Для роботи даної інформаційної системи база даних містить три зв'язані таблиці. Розроблена схема бази даних, яка використана для збереження вихідних даних системи.

Для бази даних представленої системи визначені унікальні первинні ключі та зовнішні ключі.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Люди які розплановують свій час часто користуються різними методами та способами цього планування, вони використовують різноманітні програми та застосування, починаючи з папірців на холодильнику і завершуючи різноманітними текстовими редакторами та різними щоденниками. Але суть процесу планування завжди одна, людина сама розміщує свої задачі в певному порядку, інколи розкидуючи їх по різним дням. Зазвичай, для такого планування витрачається багато часу, адже кожну задачу потрібно вносити наперед, вручну, що не є ефективним.

Так для планування свого робочого часу, скажімо, на 2 тижні потрібно витратити близько години. Також при такому плануванні рідко враховуються такі фактори як: час на перерви, протяжність цих перерв, час на те щоб «переналаштувати» свій розум на нову задачу, адже це може викликати потреби «вникати» у проблему заново. Саме тому набагато ефективніше розпланувати час так, щоб виконання 1 задачі завершувалося або на перерву, апо по завершенню робочого часу.

Також слід враховувати усі «дедлайни» для кожної з задач, що в свою чергу теж потребує час на осмислення.

Не слід забувати про те ,що якщо людина працює на себе і є так званим «фрилансером», або має незвичний графік, наприклад 3 робочі години зранку та 5 в вечері то розпланувати задачі для неї стає ще складнішим завданням, і не рідко що саме в таких ситуаціях виникають потреби працювати понаднормово, що може погано сказатись на здоров’ї людини.

Саме при плануванні гнучких графіків виникають ситуації коли потрібно максимально ефективно розпланувати свій час, враховуючи усі можливі ситуації що можуть завадити виконанню роботи. Набагато краще коли в

щоденнику стоїть не просто потрібно зробити ще 5 проектів за 15 днів, а розписано що конкретно потрібно робити щодня, для виключення ситуації коли потрібно зробити всі ті самі 5 проектів, але вже за 2 дні. Адже є багато досліджень на основі яких можна складати зручні розклади робочого часу.

3.2 Математична постановка задачі

3.2.1Присвоэння задачам пріоритетів

Оскільки немає сенсу змушуватии користувача власноруч встановлювати наскільки високий пріоритет має кожна з задач, тож будемо вважати що він у всіх задач високий, але користувач матиме змогу його знизити.

При введенні інформації про кінцеві терміни задачі користувач також буде вносити інформацію про те що ця задача є надважливою, або не є важливою, якщо такої інформації внесено не було, то вона вважається важливою.

В матриці пріоритетів Ейзенхауера є 4 сектори («важливі й термінові», «неважливі й термінові», «важливі й нетермінові», «неважливі й не термінові»), але використовуються в нашій программі з них лише 3 з них: «важливі й термінові», «неважливі й термінові», «важливі й нетермінові». Оскільки сегмент «неважливі й не термінові», не доцільно вносити до розкладу робочого часу, так як робочих задач він в собі по означенню не несе. Він заповнений різними діями накшталт поговорити з другом по телефону, пограти в ігри тощо.., і не несе в собі ніякого ефективного навантаження.[5]

Отже пріоритет задачі та її положення в матриці розраховується за формулою:

$$P=\alpha+\beta$$

,де:

P – числовий еквівалент пріоритету задачі;

α – числове значення що відображає еквівалент терміновості виконання.

β – числове значення що відображає пріоритет виконання задачі введений користувачем (важливі 0.5, середні 0.25, неважливі 0.1).

Важливість визначається на основі інформації внесеної користувачем, і має 3 рівні. По замовчуванню всі задачі важливі, але користувач може додатково позначити задачу як середню, або не важливу (якщо це наприклад його хобі).

Терміновість виконання залежить від часу до «дедлайну» задачі, що ближче термін здачі, то вища терміновість, відповідно, що далі термін здачі – то терміновість нижча.

Терміновість виконання розраховується відповідно до вище сказаного по формулі:

$$\alpha=\frac{time}{endtime}$$

,де:

endtime – це кількість робочих годин що залишається на даний момент до дедлайну задачі.

time – це час що витрачається на виконання задачі, в годинах.

Саме на основі цих розрахунків визначається пріоритет задач при складанні розкладу, в першу чергу йдуть задачі що належать до групи «важливі й термінові», до цієї групи входять не більше 2-3 задач, і вони стоять на першому місці при складанні розпорядку. Далі йдуть задачі виконання яких термінове, але не обов’язкове. І останніми в розкладі йдуть задачі що на данний момент не є терміновими але являються важливими.

Розклад складається для кожного дня окремо, тож терміновість задач зростає, і по мірі завершення одних завдань до групи «важливі й термінові» будуть потрапляти інші.

3.2.2Задача перевірки початкових термінів

При введенні користувачем певного набору задач, слід перевіряти чи реально взагалі їх виконати в задані строки, для цього в клієнтському додатку було реалізовано алгоритм перевірки валідності введених задач.

Якщо задачі неможливо виконати в задані строки можна збільшити час робочого дня, або видалити менш значимі задачі. Алгоритм виглядає наступним чином:

Крок 0. ВИЗНАЧИТИ час до кінця терміну планування.

Крок 1. ВИЗНАЧИТИ час що витрачається на кожну з задач.

Крок 2. ПОРАХУВАТИ загальний час що потрібен на виконання усіх поставлених задач.

Крок 3. ПОРІВНЯТИ результати кроку 0 та кроку 2, якщо результат 2 кроку більший за результат кроку 0, перейти до кроку 4, інкаше завершити перевірку.

Крок 4. ВИВЕСТИ повідомлення про похибку в загальних строках планування.

3.2.3Задача генерації розпорядку

Основний алгоритм в програмі, це алгоритм що відповідає за генерування розкладів. Він по суті і є основною частиною програми. Алгоритм було розроблено повністю з нуля.

Крок 0. ПЕРЕВІРИТИ чи не відповідає поточний день планування вихідному дню що задав користувач.

Крок 1. ОБЧИСЛИТИ пріоритет для кожної задачі.

Крок 2. ВИБРАТИ зі списку задач, ті що мають найвищий пріоритет.

Крок 2.1 ПЕРЕВІРИТИ чи може задача розриватись

Крок 2.2 ЯКЩО ТАК перейти до кроку 3, ІНАКШЕ записати задачу повністю

Крок 2.3 ВИДАЛИТИ задачу зі списку

Крок 3. ДОДАТИ задачу до розпорядку, час на задачу 1 година.

Крок 4. ВІДНЯТИ від загального часу на виконання роботи час що було внесено до розпорядку (-1 година).

Крок 5. ПОВТОРИТИ крок 1.

Крок 6. ПОВТОРЮВАТИ кроки 2-5 до завершення робочого часу на день.

Крок 7. ЗГРУПУВАТИ час на виконання кожної задачі в блоки (якщо є повтори).

Крок 8. ДОДАТИ в розпорядок перерви

Крок 9. ПОВТОРИТИ кроки 0 - 8 для кожного дня планування

3.3 Обґрунтування методу розв’язання

Для розв’язання поставлених задач було прийнято рішення використовувати саме наведені вище алгоритми, оскільни вони повністю відповідають поставленим цілям.

Системи пріоритетів та попередньої перевірки дозволять створити розклад, який буде повністю задовольняти потреби користувача. Задачі з найвищим пріоритетом будуть виконані максимально швидко, і при цьому задачі з середнім та низьким пріоритетами також будуть виконані до своїх дедлайнів, такий розклад забезпечить комфортну роботу користувача, та ефективно використає його робочий час.

3.4 Опис методів розв’язання

Для розв’язання подібних задач існує ціла група методів що відповідає за створення розкладів. Це теорія розкладів. Яка в свою чергу являється частиною дослідження операцій. Теорія розкладів відповідає за дослідження

задач, де потрібно впорядкування або визначення послідовності виконання сукупності задач, використання деяких засобів, тощо.

В теорії розкладів зазвичай розглядають задачі для яких попередньо розв’язано всі питання, щодо того як мають бути виконані задачі.

Як бачимо саме методи теорії розкладів було застосовано при створенні алгоритмічного забезпечення даного програмного продукту.

Висновок до розділу

У даному розділі описані математична та змістовна постановки задачі. Було наведено та детально описано основні алгоритми за допомогою яких функціонує система. Були розглянуті існуючі варіанти та методи розв’язку поставленої задачі. Було обґрунтовано вибір конкретного методу розвьязання поставленої задачі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для розробки клієнт-серверної аналітичної програми з управління робочим часом було використано наступні засоби розробки програмного забезпечення: Visual Studio 2017, .NET Framework 4.7.

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight [6].

Microsoft .NET (читається дот-нет) — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з ідей .NET є сумісність служб, написаних різними мовами.

Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET — крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell).

Захист авторських прав відноситься до створення середовищ виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET випускаються багатьма фірмами для різних мов вільно.[7]

4.2 Вимоги до технічного забезпечення

Конфігурація засобів, що визначається виходячи із можливості їх забезпечити запуск клієнт-серверної аналітичної програми з управління робочим часом.

Для правильної роботи програми потрібен компь’ютер з наведеною нижче конфігурацією:

- процесор з частотою не нижче 2 ГГц (Intel Core 2 Duo, Athlon 64 X2);
- оперативну пам’ять, не менше 2 Гб.

Додатково має бути встановлено таке програмне забезпечення:

- операційна система Windows 10 64-bit/32- bit;
- .NET Framework 4.7.

4.2.1Загальні вимоги

Для правильного функціонування програми потрібен комп’ютер з інтернет з’єднанням. Також не слід забувати про те що потрібен комп’ютер з вище описаними системними вимогами, та монітор, з маніпуляторами клавіатурою та мишкою.

4.3 Архітектура програмного забезпечення

Клієнт-серверна архітектура, це архітектура в якій завдання розподілені між постачальниками послуг, так званими серверами та замовниками послуг, що називаються клієнтами.

Фактично клієнт і сервер - це програмні продукти, або програмне забезпечення. Зазвичайці програми розташовані на різних обчислювальних машинах (комп'ютерах) і взаємодіють між собою через мережу, за допомогою мережевих протоколів, але вони можуть бути розташовані й на одній обчислювальній машині.

Програми-сервери очікують від програм-клієнтів запити і передають їм свої ресурси в вигляді данних (наприклад завантаження данних, робота з базами данних, тощо) або в вигляді сервісних функцій (наприклад робота з електронною поштою, спілкування через системи миттєвого обміну повідомленнями або перегляд веб-сторінок).

Оскільки одна программа сервер має обробляти запити багатьох програм-клієнтів, то її розміщують зазвичай на спеціально виділеній обчислювальній машині, що налаштована особливим способом, зазвичай на машині-сервері можу розміщуватись не одна програма-сервер, а отже і обчислювальні потужності такої машини мають бути вісокими.

Переваги клієнт серверної архітектури:

- відсутність дублювання коду програми-сервера, програмами клієнтами, що в свою чергу дозволяє зменшити розміри останніх;
- оскільки всі обчислювальні операції відбуваються на сервері, то вимоги до клієнтської машини, на якій встановлено клієнт можуть бути знижені, в порівнянні з ситуацією коли ми маємо одну комплексну програму;
- всі данні клієнтів зберігаються на сервері, який зазвичай має кращий захист ніж більшість клієнтських комп’ютерів;
- можна вносити корективи в розрахунки системи, не примушуючи клієнтів обновлювати свої програми.

Для розробки програмного продукту було вирішено використовувати саме клієнт серверну архітектуру, оскільки це дозволить вносити корективи в систему генерації розкладів не завдаючи клієнтам зайвих труднощів. Також дана архітектура дозволить ефективно збирати статистичні дані з використання програми, на основі яких можна буде ефективно та оперативно вносити корективи в роботу системи. Не менш важливим фактором є те, що

клієнт-серверна архітектура дозволить максимально знизити навантаження на систему користувача, та перекласти усі ці обов'язки на сторону сервера.

4.3.1Діаграма класів

Як ми бачимо зі схеми структурної класів, що наведена в графічних матеріалах, в клієнтському додатку ми маємо клас «sender», в ньому реалізовано функції що відповідають за відправку інформації на сервер. Також в клієнтському додатку є багато класів-інтерфейсів що відповідають за взаємодію користувача й клієнтського додатку.

В серверній частині додатку можна спостерігати клас «work», в якому було реалізовано роботу алгоритму генерації програми.

4.3.2Діаграма послідовності

На схумах структурних послідовності, що наведені в графічних матеріалах для клієнтського та сервеного додатків, наведено реалізацію роботи алгоритмів в програмі. Також наведено схему взаємодії в програмі.

4.3.3Діаграма компонентів

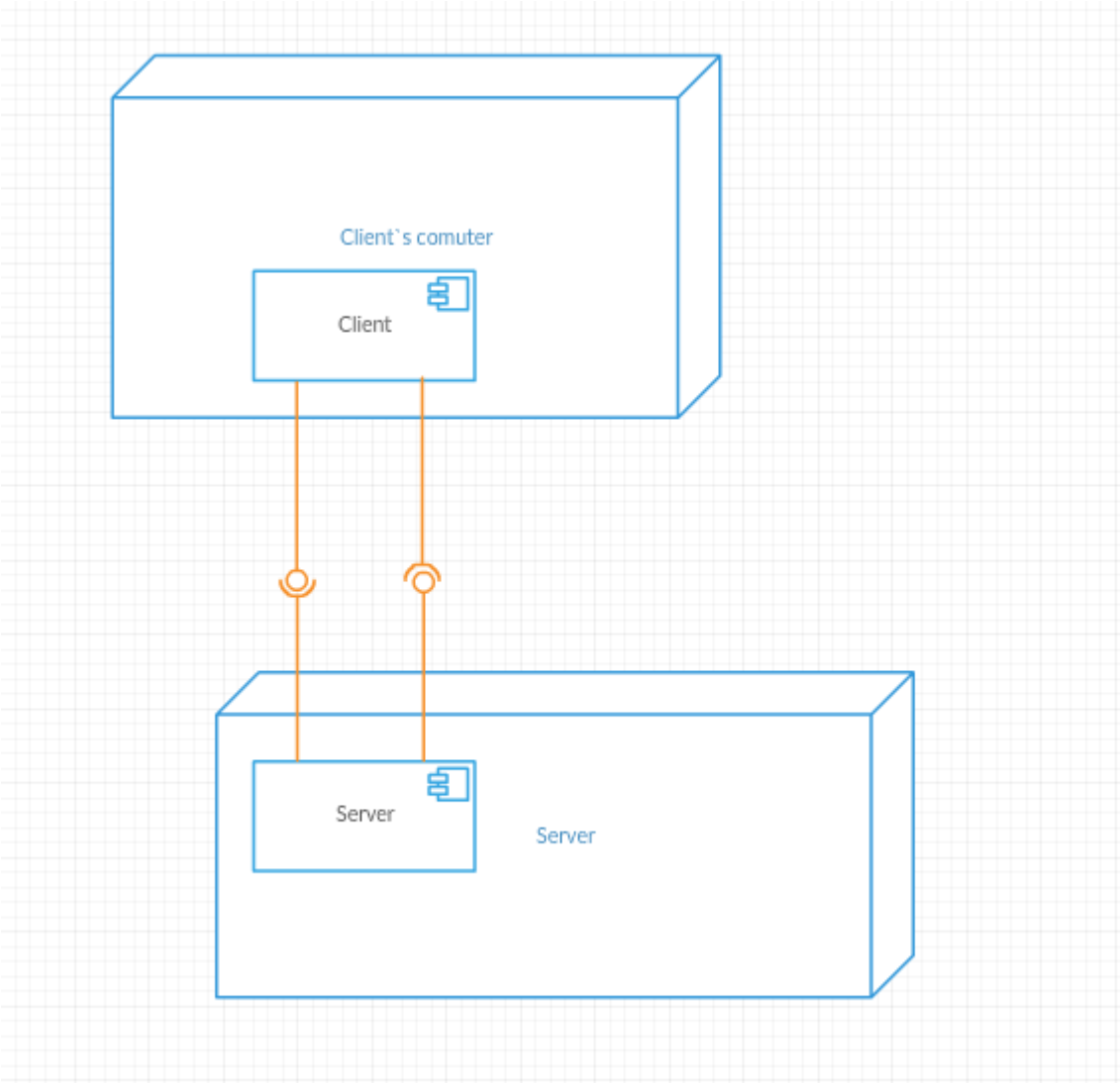


Рисунок 4.1 – Схема структурна компонентів

Як видно з наведеної вище діаграми клієнт серверна аналітична система з управління робочим часом складається з двох машин. Клієнтська частина встановлена на комп’ютері клієнта, а серверна на комп’ютері що виділено під сервер. Вони здійснюють зв’язок між собою через мережу інтернет.

4.3.4Специфікація функцій

В таблицях 4.1-4.2 наведено специфікації функцій що використовуються в системі.

Таблиця 4.1 – «Клієнтська частина»

Назва функції	Опис функції
button_Close();	Завершення роботи програми
Enter_Click();	Відповідає за обробку натиснення кнопки «Війти», та за результати обміну інформацією з сервером
Register button2_Click();	Відповідає за обробку натиснення кнопки «Зареєструватись», та за результати обміну інформацією з сервером
Exit_Click();	Завершення роботи програми
Calendar button_Click();	Перевірка на відкриття, та запуск форми що відповідає за відкриття форми календар
org_Click();	Перевірка на відкриття, та запуск форми що відповідає за відкриття форми блокнот для записів
CreateTT button_Click();	Перевірка на відкриття, та запуск форми що відповідає за відкриття форми створення розкладу
T_T_Click();	Перевірка на відкриття, та запуск форми що відповідає за відкриття форми поточний розклад
HTT_Click();	Перевірка на відкриття, та запуск форми що відповідає за відкриття форми історія розкладів
ST_Click();	Перевірка на відкриття, та запуск форми що відповідає за відкриття форми налаштування
INF_Click();	Перевірка на відкриття, та запуск форми що відповідає за відкриття форми інформації
CurrentTime();	Повертає поточний час

Selected TimeZone();	Реалізує обробку вибору часової зони
UpdateTime();	Відповідає за оновлення часу в формі

Продовження таблиці 4.1

Назва функції	Опис функції
TimeZones();	Реалізує повернення часу з часової зони
back_Click();	Функція що реалізує обробку натиснення кнопки назад
checkBox _Checked();	Функція що відслідковує зміну чекбоксу
checkBox _Unchecked();	Функція що відслідковує зміну чекбоксу
button1_Click();	Функція що відповідає за обробку кнопки збереження, що дозволяє подальше введення інформації
SaveTask();	Функція що реалізує збереження задачі
Save_Click();	Функція що реалізує збереження функціїї зберегти
Test_All();	Функція що реалізує перевірку всіх введених данних
SV_Click();	Функція збереження
button2_Click();	Функція розблокування елементів введення
Workd();	Функція що перевіряє реальність роботи
Window _Closed();	Функція що відповідає за обробку закриття вікна
Write_in_txt();	Функція що відповідає підготовку до запису в текстовий файл
Write();	Функція що реалізує безпосередньо запис інформації в файл
Window _Loaded();	Функція що відповідає за завантаження вікна

Продовження таблиці 4.1

Назва функції	Опис функції
send();	Функція що відправляє зібрану інформацію про розклад на сервер
sendL();	Функція що відправляє інформацію що відповідає за вхід на сервер

Таблиця 4.2 – «Серверна частина»

Назва функції	Опис функції
logining();	Функція що відповідає за обробку отриманих від клієнта даних
parse();	Функція що відповідає за парсинг інформації про розклад
Format();	Функція що відповідає за форматування отриманої інформації
Workd();	Функція що відповідає за роботу алгоритму
prior_sort();	Функція що відповідає за обробку пріоритетів
data_sort();	Функція що відповідає за обробку даних
prio_k();	Функція що відповідає за генерацію коефіцієнтів
grup();	Функція що відповідає за групування елементів
writeintt();	Функція що відповідає за запис інформації в базу даних
SaveTT();	Функція що відповідає за збереження інформації в базі даних

SaveTTpart2();	Функція що відповідає за збереження інформації в базі даних
GetConnection String();	Функція що відповідає за повернення строки підключення до бази даних

4.4 Опис звітів

В якості звіту клієнт серверна аналітична програма повертає HTML файл розкладу. Файл складається з розкладу на кожен день, з інформацією про задачі на кожен день планування.

Висновок до розділу

В даному розділі було розглянуто структуру програми. Також було описано усі основні функції клієнтської частини програми та серверної частини. Були відображені та прокоментовані діаграми послідовностей, компонентів та класів.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

В ході розробки клієнт-серверної аналітичної системи з управління робочим часом було вирішено усі поставлені задачі.

В першу чергу користувач бачить вікно входу до системи (рисунок 5.1). Користувач може зареєструватись у системі ввівши логін, довжиною від 2 до 6 символів, та натиснувши кнопку «зареєструватись». Якщо користувач уже був зареєстрований у системі то він може війти під своїм логіном, ввівши логін та пароль і натиснувши кнопку «вхід». Якщо користувач вже зареєстрований в системі, при спробі зареєструватись він отримає повідомлення, що користувач з таким іменем уже зареєстрований. При вході або реєстрації користувач отримає відповідні повідомлення (рисунок 5.2 та рисунок 5.3).

Рисунок 5.1 – Форма реєстрації

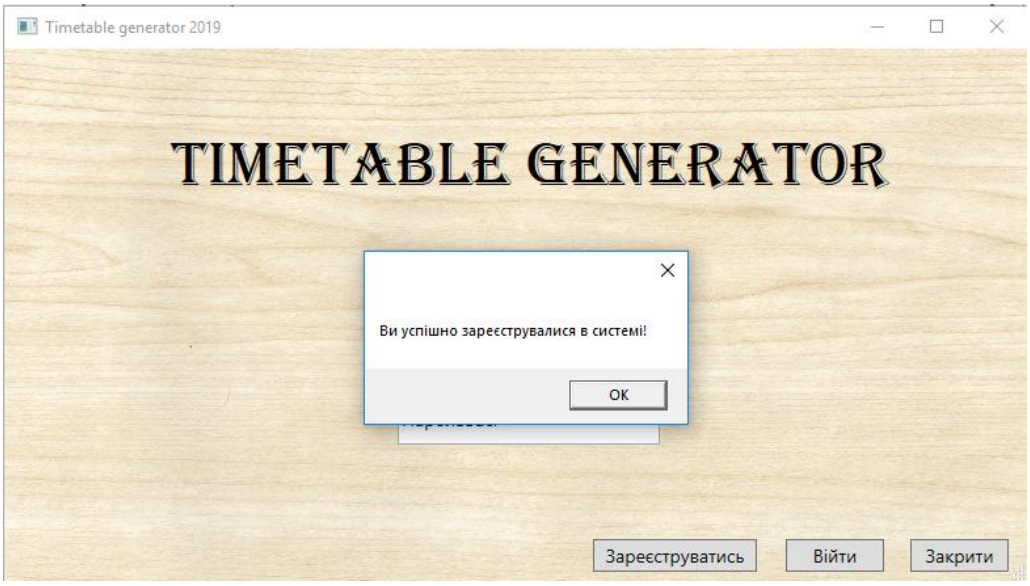


Рисунок 5.2 – Повідомлення про успішну реєстрацію

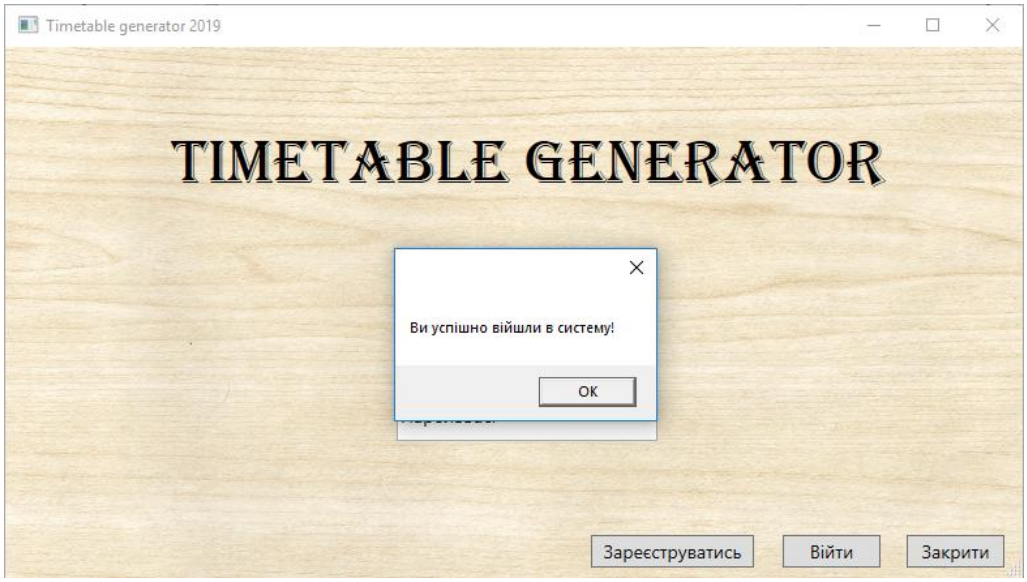


Рисунок 5.3 – Повідомлення про успішний вхід

Далі користувач потрапляє на форму меню (рисунок 5.4). Саме з даної форми користувач отримує доступ до всього функціоналу програми, а саме до наступних форм: календар, блокнот для записів, складання розкладу, поточний розклад, історія розкладів, налаштування, інформація.

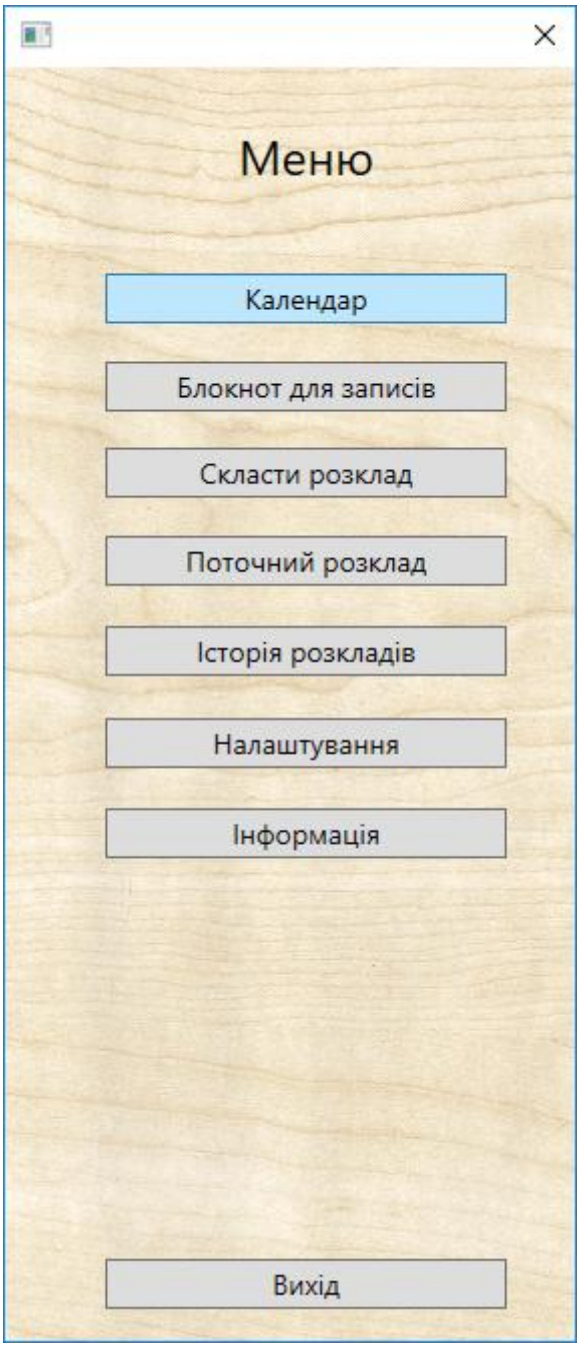


Рисунок 5.4 – Основна форма меню

При натискані кнопки «календар», відкривається форма з поточним часом та календарем (рисунок 5.5), її можна використовувати при складанні розкладу а також при записах в форму для записів. Додатковим функціоналом являється можливість переглянути поточний час в будь якому місці світу.

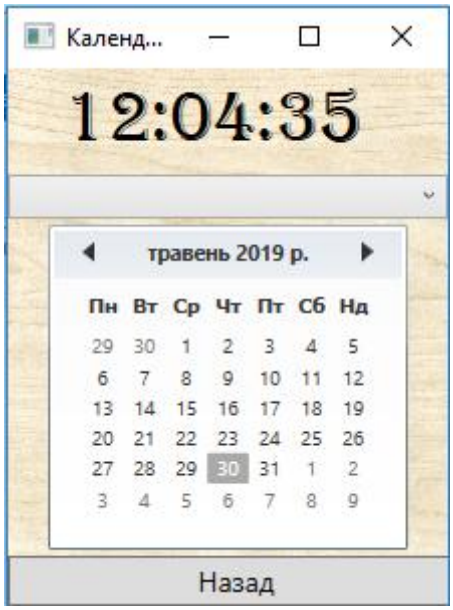


Рисунок 5.5 – Форма «Календар»

При натискані кнопки «Блокнот для записів», відкривається форма що відповідає за запис додаткової інформації (рисунок - 5.6), що не входить до поточного розкладу. Інформація внесена в блокнот для записів зберігається на локальній машині користувача.

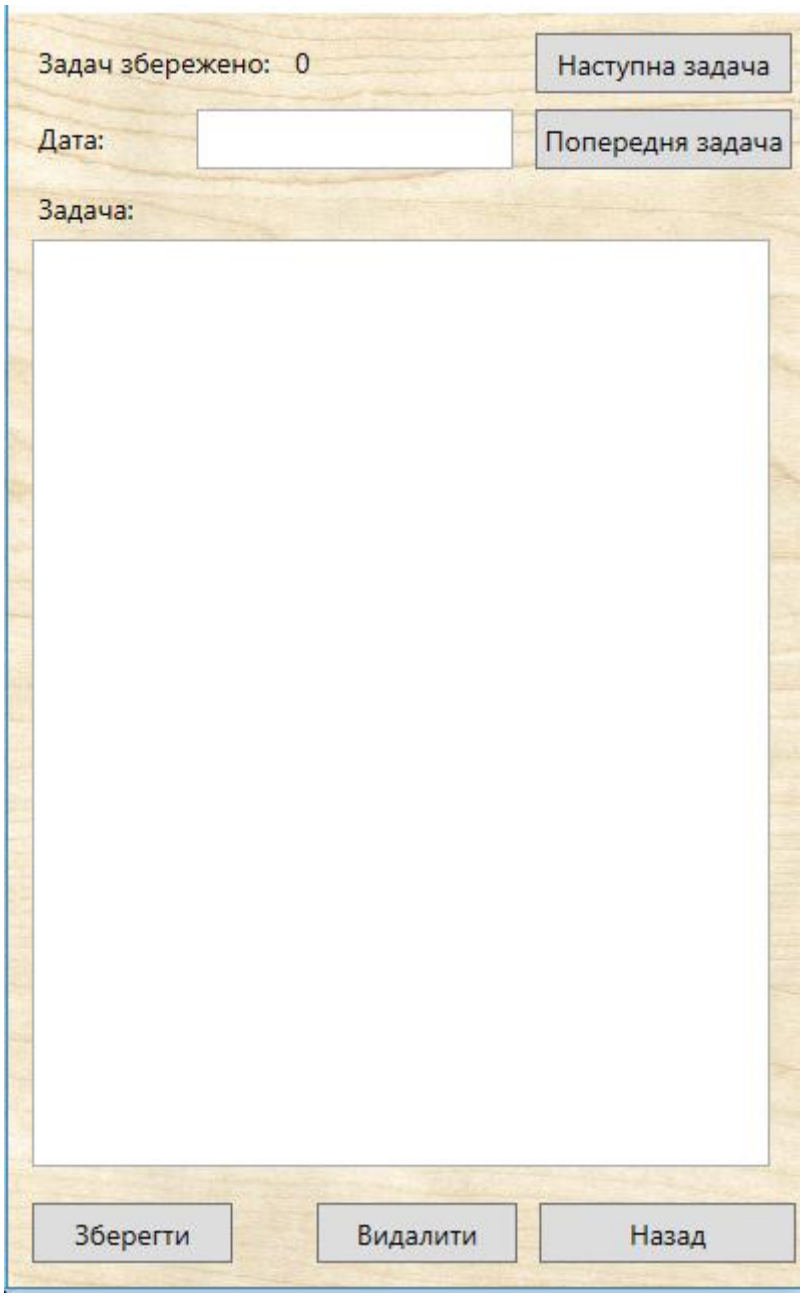


Рисунок 5.6 – Форма «Блокнот для записів»

При натисканні кнопки «скласти розклад», відкривається форма «Створення розкладу»(рисунок 5.7). На ній користувач вносить всю потрібну для створення розкладу інформацію.

Спершу він заповнює верхню частину форми. Вводить інформацію про загальні дати планування. Воодить інформацію про довжину робочого дня. Також відмічає дні що вважатимуться вихідними.

Далі користувач натискає кнопку «Зберегти». Верхня частина форми стає неактивною, а нижня активною. Користувач заповнює інформацію про конкретну задачу, встановлює їй пріоритет, після чого натискає кнопку «зберегти задачу». Якщо вся інформація введена коректно користувач отримує відповідне повідомлення (рисунок 5.8). Після внесення всіх задач, користувач натискає кнопку «Зберегти розклад», та вся інформація відправляється на обробку на сервері, а користувач отримує відповідне повідомлення (рисунок 5.9).

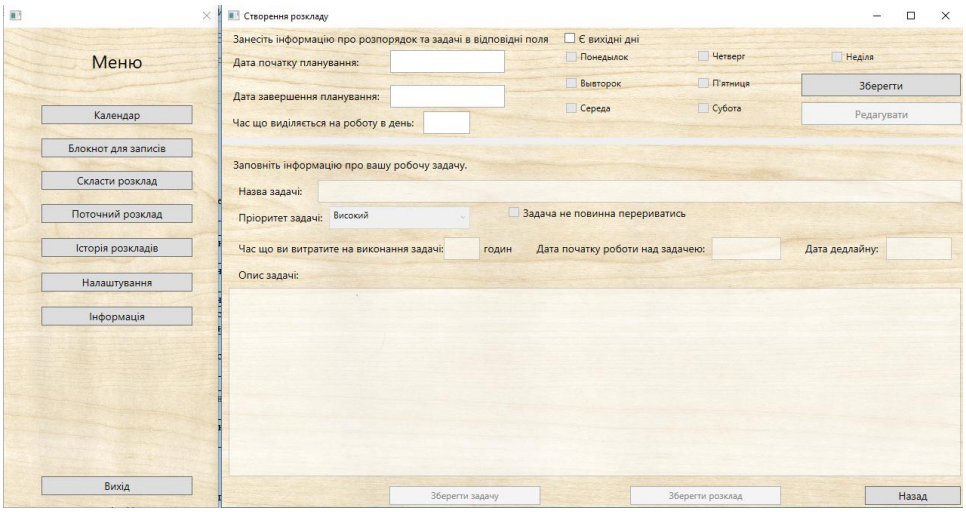


Рисунок 5.7 – Форма «Створення розкладу»

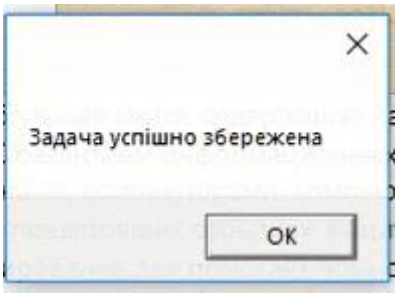


Рисунок 5.8 – Повідомлення про успішне збереження

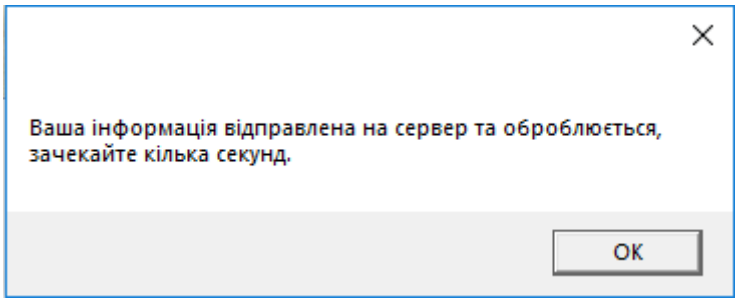


Рисунок 5.9 – Повідомлення про відправку на сервер

При натисканні кнопки «поточний розклад», якщо розклад було попередньо згенеровано відкривається форма «поточний розклад» (рисунок 5.10). В ній користувач зможе переглянути усі внесені раніше задачі, згенерований розклад, та в разі потреби зможе їх скорегувати.

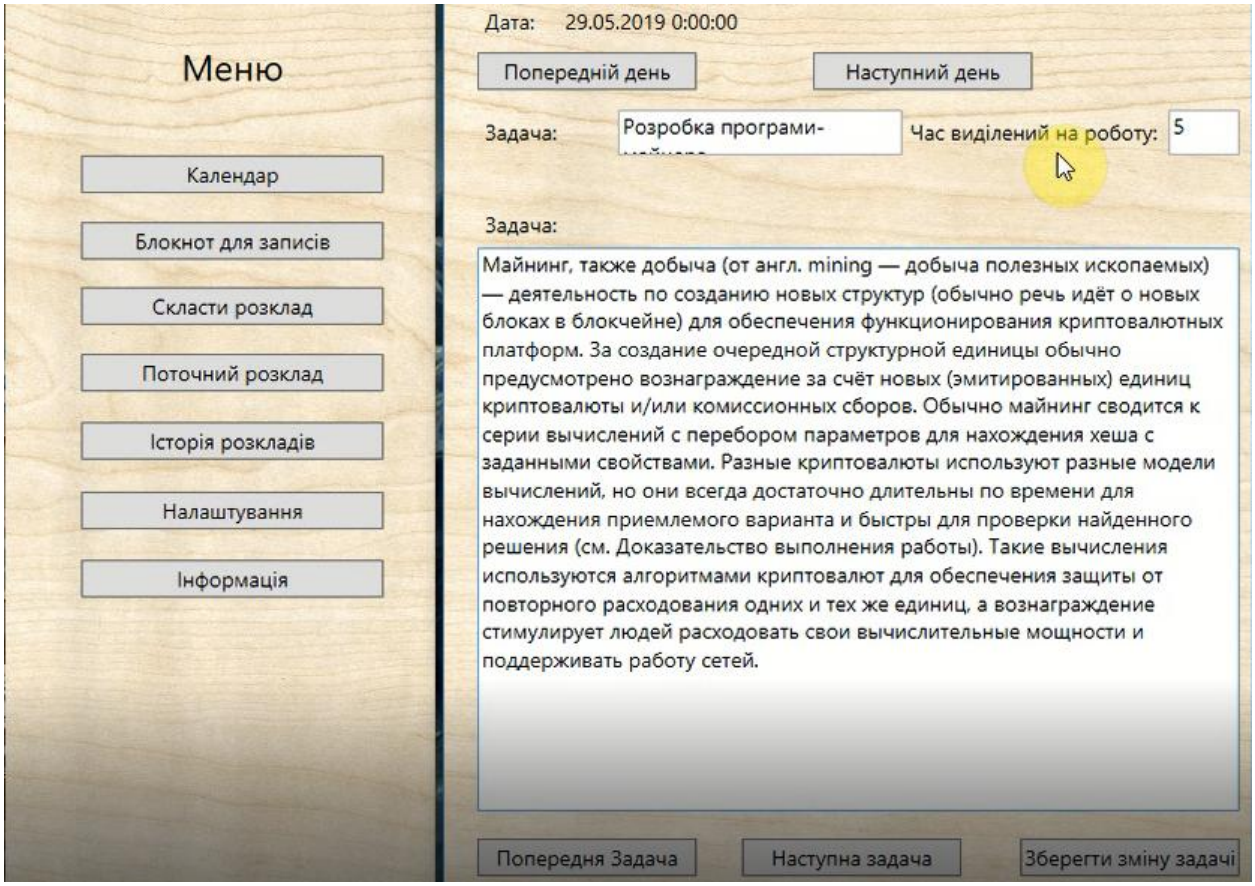


Рисунок 5.10 – Форма «Поточний розклад»

При натисканні клавіші «історія розкладів» користувач зможе завантажити з сервера згенеровані раніше файли розкладів.

При натисканні клавіші «Налаштування» користувач зможе потрапити в меню налаштувань, де зможе змінити за бажанням колір фону.

При натисканні клавіші «Інформація» користувач зможе переглянути усю наявну інформацію про роботу додатку, та про розробника.

Результатом роботи програми є генерація HTML документу, наведеного на рисунку 5.11.

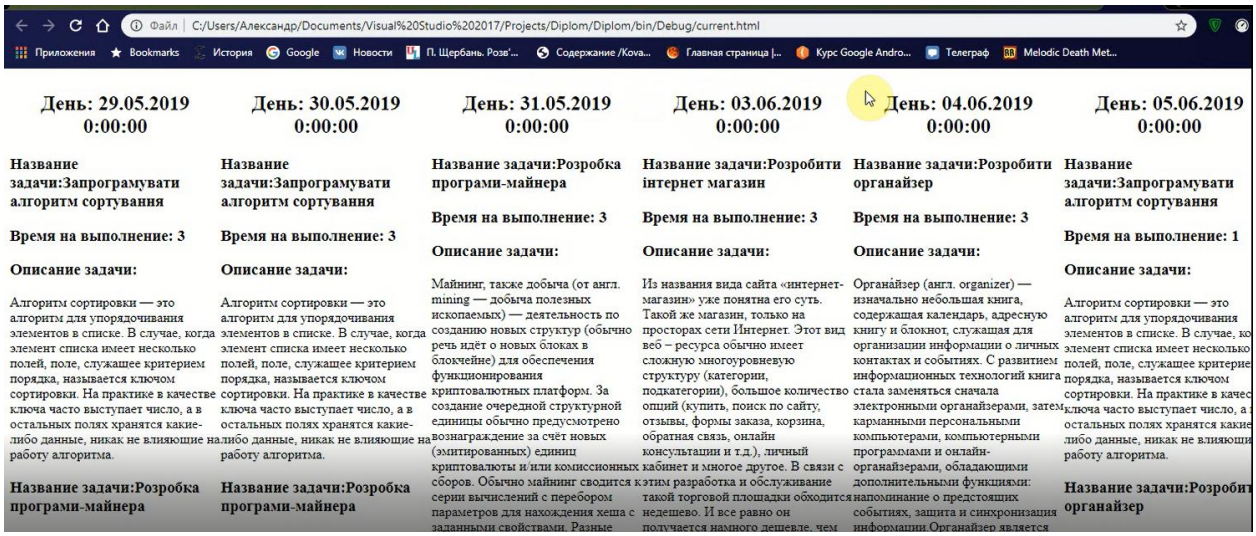


Рисунок 5.11 – Файл розкладу

5.2 Випробування програмного продукту

Таблиця 5.1 – Перевірка підключення до сервера зі сторони клієнта

Мета тесту:	Перевірка функції «Підключення(Клієнт)»
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі.
Очікуваний результат:	Клієнтська програма запущена; При введенні логіну та паролю з’являється відповідне повідомлення
Вхідні дані:	Логін та пароль
Схема проведення тесту:	Після запуску клієнтського додатку ввести інформацію в поля логін та пароль; Натиснути кнопку «Зареєструватись»;

Таблиця 5.2 – Перевірка відправки інформації клієнтом

Мета тесту:	Перевірка відправки інформації клієнтом
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Набір певних задач.
Схема проведення тесту:	Ввести в програмі клієнта набір задач, та натиснути кнопку згенерувати розклад.
Очікуваний результат:	На серверній програмі, в панелі інформації з’явиться інформація про отримання від клієнту початкових даних для генерації розпорядку

Таблиця 6.3 – Перевірка планування зі сторони сервера

Мета тесту:	Перевірка функції «Планування(сервер)»
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Інформація з клієнту
Схема проведення тесту:	Здійснити тест описаний у 6.2;
Очікуваний результат:	На серверній програмі, в панелі інформації з’явиться інформація про те що згенеровано новий розклад для клієнта, при натисненні на кнопку «деталі», можна буде його переглянути

Таблиця 5.4 – Перевірка відправки зі сторони сервер

Мета тесту:	Перевірка функції «Відправка (Сервер)»
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у

	стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Виконати дії з тесту 6.2.
Схема проведення тесту:	Дочекатись надходження повідомлення від серверу з згенерованим розкладом;
Очікуваний результат:	З’явиться відповідне повідомлення, про те що ваш розклад згенеровано

Таблиця 5.5 – Перевірка функції «Convert» (клієнт-сервер)

Мета тесту:	Перевірка функції «Convert» (клієнт-сервер)
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Інформація з клієнта.
Схема проведення тесту:	Виконати дії тесту 6.2 Дочекатись надходження інформації від клієнта;

Очікуваний результат:	При перегляді вхідних даних на сервер немає різниці між інформацією введеною в клієнт

Таблиця 5.6 – Перевірка функції «Convert» (сервер клієнт)

Мета тесту:	Перевірка функції «Convert» (сервер клієнт)
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Виконати дії тесту 6.2
Схема проведення тесту:	Дочекатись надходження інформації з серверу.
Очікуваний результат:	Розклад що було згенеровано на сервері не відрізняється від розкладу що відображається на клієнті

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій

комплексу задач клієнт-серверної аналітичної програми з управління робочим часом вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

В ході комплексу випробувань не було виявлено ніяких помилок в ході тестування клієнт-сервеної аналітичної програми з управління робочим часом.

Висновок до розділу

В даному розділі описано керівництво користувача. Наведено опис основних функцій програми, та продемонстровано роботу програми на прикладі скриншотів. Було наведено опис тестів програми, та описано результати тестування.

ЗАГАЛЬНІ ВИСНОВКИ

В даній роботі було реалізовано клієнт серверну аналітичну програму з управління робочим часом користувача. Було реалізовано весь спектр задач що були потрібні для досягнення мети проекту.

Був проведений опис предметного середовища, було визначено вимоги, було знайдено аналоги розроблюваного програмного продукту та зроблено порівняльну характеристику, а також було визначено призначення розробки, цілі та задачі що потрібно вирішити для досягнення поставлених цілей. Виходячи з висновків порівняння з аналогами можемо зробити висновок, що наше застосування буде корисним, та ефективнішим за аналоги.

Надано описи структури таблиць бази даних та детальну інформацію по кожній сутності з описом полів. Для роботи даної інформаційної системи база даних містить три зв’язані таблиці. Розроблена схема бази даних, яка використана для збереження вихідних даних системи. Для бази даних представленої системи визначені унікальні первинні ключі та зовнішні ключі.

Описані математична та змістовна постановки задачі. Було наведено та детально описано основні алгоритми за допомогою яких функціонує система. Були розглянуті існуючі варіанти та методи розв’язку поставленої задачі. Було обґрунтовано вибір конкретного методу розв’язання поставленої задачі. Для розробки використовувалися засоби розробки середовища Visual Studio 2017.

Описано керівництво користувача. Наведено опис основних функцій програми, та продемонстровано роботу програми на прикладі скриншотів. Було наведено опис тестів програми, та описано результати тестування.

ПЕРЕЛІК ПОСИЛАНЬ-

1. ITteach.ru [Електронний ресурс] / ITteach. – 2019. - Режим доступу до ресурсу: https://itteach.ru/bpwin/metodologiya-idef0.

2. Jetbrains. Products [Електронний ресурс] / JetBrains. – 2019. – Режим доступу до ресурсу: https://jetbrains.ru/products/youtrack/.

3. LeaderTask[Електронний ресурс] / Organizer LeaderTask – 2019. – Режим доступу до ресурсу: https://www.leadertask.com/.

4. Бесплатный органайзер Exiland Assistant Free [Електронний ресурс] / Exiland Software. – 2019. – Режим доступу до ресурсу:https://exiland-soft.com/ru/besplatniy -organizer.html

5. Матриця Ейзенхауера для складання списку справ. [Електронний ресурс] – 2019. – Режим доступу до ресурсу: http://how-to-do.org/matrytsa-ejzenhauera/.

6. Visual Studio [Електронний ресурс] / Microsoft – 2019. - Режим доступу до ресурсу: https://visualstudio.microsoft.com/ru/?rr=https%3A%2F%2Fru.wikipedia.org%2F

7. Microsoft .NET [Електронний ресурс] / Microsoft – 2019. - Режим доступу до ресурсу: https://dotnet.microsoft.com/

Додаток А

Тексти програмного коду

Клієнт-серверна аналітична система управління робочим часом

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 арк

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

```
namespace DiplomServer
{
   class workclass
   {
      static int port = 8005;
      static int priority = 0;
      static string input;
      private static string n;
      public static int idTT;
      public static string FullTT;
      private static string[] Time = new string[4];
      public static double pr;
      private static string[,] TimeTable = new string[1000, 10];
      private static string[,] Tasks = new string[1000, 10];
      static string[,] TimeTable2;

      public static void working()
      {
         input = "[zadacha1*Високий*15*1.1.2015* 9.1.2015*й1й1й1й1*1*-6-
7^zadacha2*Високий*10*2.1.2015* 9.1.2015*й2й2й2й2*1*-6-
7^zadacha3*Високий*20*1.1.2015* 9.1.2015*й3й3й3й3*1*-6-
7^10.1.2015*1.1.2015*10]^|timetable|2";
         parse();
         grup();
         SaveTT();
      }

      public static void logining()
      {
         input = "";
```

```
        IPEndPoint ipPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), port);

        // создаем сокет
        Socket listenSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        try
        {
            // связываем сокет с локальной точкой, по которой будем принимать данные
            listenSocket.Bind(ipPoint);

            // начинаем прослушивание
            listenSocket.Listen(10);

            while (true)
            {
        string ansv = "";
                Socket handler = listenSocket.Accept();
                // получаем сообщение
                StringBuilder builder = new StringBuilder();
                int bytes = 0; // количество полученных байтов
                byte[] data = new byte[2000000]; // буфер для получаемых данных

                do
                {
                    bytes = handler.Receive(data);
                    builder.Append(Encoding.UTF8.GetString(data, 0, bytes));
                }
                while (handler.Available > 0);
                // System.Windows.MessageBox.Show(DateTime.Now.ToShortTimeString() + ": " + builder.ToString());
                input = builder.ToString();
```

```
//Table1TableAdapter a = new Table1TableAdapter();
SqlConnection sqlConn = new SqlConnection(GetConnectionString());
sqlConn.Open();
//выполняем запрос

string[] substrings2 = input.Split('^');
string[] substrings1 = input.Split('^');
string[] substrings3= new string[10];
bool testTT = input.Contains("|");
if(testTT==true)
{
substrings3 = input.Split('|');
}
if (testTT==true)
{
   if(substrings3[1]== "timetable")
   {
   parse();
   grup();
   ansv = FullTT;
   SaveTT();
   }
}
else
{

   SqlCommand sqlCom = new SqlCommand($"SELECT * FROM Table1
WHERE (UserLogin LIKE (N'{substrings2[0]}'))", sqlConn);
   SqlDataReader dr = sqlCom.ExecuteReader();
   //результат запроса суем в таблицу
   DataTable dt = new DataTable();
   dt.Load(dr);
   //dt.Rows.Count;
```

```
if (substrings2[2] != "reg")
{
    bool ind = false;
    if (dt.Rows.Count != 0)
    {
        int id = -1;
        for (int i = 0; i < dt.Rows.Count; i++)
        {
            string field = dt.Rows[i].Field<string>(2);
            if (field == substrings2[1])
            {
                id = i;
                ind = true;
            }
        }
        if (ind == true)
        {

            ansv = "registred" + "^" + Convert.ToString(dt.Rows[id].Field<int>(0));
        }
        else
        {
            ansv = "vrong_pass";
        }

    }
    else
    {
        ansv = "vrong_login";

    }
}
else
```

```
            {
                //#region register
                //
                bool ind = false;
                if (dt.Rows.Count != 0)
                {
                    ansv = "registred_yet";
                }
                else
                {
                    SqlConnection connection = new SqlConnection(GetConnectionString());
                        connection.Open();
                        SqlCommand command = new SqlCommand($"INSERT INTO Table1 (UserLogin,UserPass) VALUES(N'{substrings1[0]}',N'{substrings1[1]}')", connection);
                        command.ExecuteNonQuery();
                        ansv = "registred";

                    SqlConnection sqlConn1 = new SqlConnection(GetConnectionString());
                    sqlConn1.Open();
                    SqlCommand sqlCom1 = new SqlCommand($"SELECT * FROM Table1 WHERE (UserLogin LIKE (N'{substrings1[0]}'))", sqlConn1);
                    SqlDataReader redr = sqlCom1.ExecuteReader();
                    //результат запроса суем в таблицу
                    DataTable datb = new DataTable();
                    datb.Load(redr);
                    ansv += "^" + Convert.ToString(datb.Rows[0].Field<Int32>(0));
                }
            }
        }
        //SELECT* FROM[dbo].[student] WHERE(Surname LIKE(RTRIM(@Surname)+'%') OR Name LIKE(RTRIM(@Name)+'%') OR Fathername LIKE(@Fathername+'%'))
```

```
            // отправляем ответ
            //string message = "ваше сообщение доставлено";
            data = Encoding.UTF8.GetBytes(ansv);
            handler.Send(data);
            // закрываем сокет
            handler.Shutdown(SocketShutdown.Both);
            handler.Close();
        }
    }
    catch (Exception ex)
    {
        System.Windows.MessageBox.Show((ex.Message));
    }
}

public static void parse()
{
    string str_tmp = input;
    string[] sub2 = str_tmp.Split('|');
    str_tmp = sub2[0];
    str_tmp = str_tmp.Remove(0, 1);
    str_tmp = str_tmp.Remove(str_tmp.Length - 2);
    //System.Windows.MessageBox.Show(str_tmp);
    string[] substrings1 = str_tmp.Split('^');
    int i = 0;
    for (int b = 0; b < substrings1.Length - 1; b++)
    {
        string[] str = substrings1[i].Split('*');
        for (int a = 0; a < 8; a++)
        {
            TimeTable[i, a] = str[a];
        }
        i++;
```

```
        }
        priority = substrings1.Length - 1;
        string[] str1 = substrings1[i].Split('*');
        string[] str2 = input.Split('|');
        Time[0] = str1[0];
        Time[1] = str1[1];
        Time[2] = str1[2];
        Time[3] = str2[2];
        Format();
    }

    public static void Format()
    {
        for (int a = 0; a < priority; a++)
        {
            if (TimeTable[a, 1] == "Низький")
            {
                TimeTable[a, 1] = Convert.ToString(0.1);
            }
            else if (TimeTable[a, 1] == "Середній")
            {
                TimeTable[a, 1] = Convert.ToString(0.25);
            }
            else
            {
                TimeTable[a, 1] = Convert.ToString(0.5);
            }
        }

        try
        {
            n = TimeTable[0, 7];
            TimeTable2 = new string[workd(Time[1], Time[0]), Convert.ToInt32(Time[2])];
```

```
            //string[,] TimeTable2 = new string[10, 10];
            int days = Convert.ToInt32((DateTime.Parse(Time[0]) -
DateTime.Parse(Time[1])).Days + 1);
            DateTime startDate = DateTime.Parse(Time[1]);
            DateTime endDate = DateTime.Parse(Time[0]);
            DateTime current = startDate;
            int workdays = days;
            int t = 0;
            for (int i = 0; i < days; i++)
            {
               t = 0;
               current = startDate.AddDays(i);
               prio_k();
               testn();
               if (n.IndexOf("1") == -1 && DayOfWeek.Monday == current.DayOfWeek)
               {
                  do
                  {
                     int r = 0;
                     prio_k();
                     r = prior_sort();
                     if (TimeTable[r, 0] != "")
                     {
                        TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                        TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                     }
                     if (TimeTable[r, 2] == "0")
                     {
                        TimeTable[r, 0] = null;
                        TimeTable[r, 1] = null;
                        TimeTable[r, 2] = null;
                        TimeTable[r, 3] = null;
```

```
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
                    TimeTable[r, 8] = null;
                }
                prior_sort();
                t++;
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
        }
        testn();
        if (n.IndexOf("2") == -1 && DayOfWeek.Tuesday == current.DayOfWeek)
        {
            do
            {
                int r = 0;
                prio_k();
                r = prior_sort();
                if (TimeTable[r, 0] != "")
                {
                    TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                    TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                }
                if (TimeTable[r, 2] == "0")
                {
                    TimeTable[r, 0] = null;
                    TimeTable[r, 1] = null;
                    TimeTable[r, 2] = null;
                    TimeTable[r, 3] = null;
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
```

```
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
                    TimeTable[r, 8] = null;
                }
                prior_sort();
                t++;
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
        }
        testn();
        if (n.IndexOf("3") == -1 && DayOfWeek.Wednesday == current.DayOfWeek)
        {
            do
            {
                int r = 0;
                prio_k();
                r = prior_sort();
                if (TimeTable[r, 0] != "")
                {
                    TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                    TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                }
                if (TimeTable[r, 2] == "0")
                {
                    TimeTable[r, 0] = null;
                    TimeTable[r, 1] = null;
                    TimeTable[r, 2] = null;
                    TimeTable[r, 3] = null;
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
```

```
                    TimeTable[r, 8] = null;
                }
                prior_sort();
                t++;
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
        }
        testn();
        if (n.IndexOf("4") == -1 && DayOfWeek.Thursday == current.DayOfWeek)
        {
            do
            {
                int r = 0;
                prio_k();
                r = prior_sort();
                if (TimeTable[r, 0] != "")
                {
                    TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                    TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                }
                if (TimeTable[r, 2] == "0")
                {
                    TimeTable[r, 0] = null;
                    TimeTable[r, 1] = null;
                    TimeTable[r, 2] = null;
                    TimeTable[r, 3] = null;
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
                    TimeTable[r, 8] = null;
                }
```

```
                prior_sort();
                t++;
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
        }
        testn();
        if (n.IndexOf("5") == -1 && DayOfWeek.Friday == current.DayOfWeek)
        {
            do
            {
                int r = 0;
                prio_k();
                r = prior_sort();
                if (TimeTable[r, 0] != "")
                {
                    TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                    TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                }
                if (TimeTable[r, 2] == "0")
                {
                    TimeTable[r, 0] = null;
                    TimeTable[r, 1] = null;
                    TimeTable[r, 2] = null;
                    TimeTable[r, 3] = null;
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
                    TimeTable[r, 8] = null;
                }
                prior_sort();
                t++;
```

```
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
        }
        testn();
        if (n.IndexOf("6") == -1 && DayOfWeek.Saturday == current.DayOfWeek)
        {
            do
            {
                int r = 0;
                prio_k();
                r = prior_sort();
                if (TimeTable[r, 0] != "")
                {
                    TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                    TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                }
                if (TimeTable[r, 2] == "0")
                {
                    TimeTable[r, 0] = null;
                    TimeTable[r, 1] = null;
                    TimeTable[r, 2] = null;
                    TimeTable[r, 3] = null;
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
                    TimeTable[r, 8] = null;
                }
                prior_sort();
                t++;
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
```

```
            }
            testn();
            if (n.IndexOf("7") == -1 && DayOfWeek.Sunday == current.DayOfWeek)
            {
               do
               {
                  int r = 0;
                  prio_k();
                  r = prior_sort();
                  if (TimeTable[r, 0] != "")
                  {
                     TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                     TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                  }

                  if (TimeTable[r, 2] == "0")
                  {
                     TimeTable[r, 0] = null;
                     TimeTable[r, 1] = null;
                     TimeTable[r, 2] = null;
                     TimeTable[r, 3] = null;
                     TimeTable[r, 4] = null;
                     TimeTable[r, 5] = null;
                     TimeTable[r, 6] = null;
                     TimeTable[r, 7] = null;
                     TimeTable[r, 8] = null;
                  }
                  prior_sort();
                  t++;
               } while (t < Convert.ToInt32(Time[2]));
               t = 0;
            }
```

```
        }
    }
    catch (Exception)
    {

    }


}



private static int workd(string dd1, string dd2)
{
    try
    {
        DateTime startDate = DateTime.Parse(dd1);
        DateTime endDate = DateTime.Parse(dd2);
        DateTime current = startDate;
        int days = (endDate - startDate).Days + 1;
        int workdays = days;
        for (int i = 0; i < days; i++)
        {
            testn();
            current = startDate.AddDays(i);
            if (n.IndexOf("1") > -1 && DayOfWeek.Monday == current.DayOfWeek)
            {
                workdays += -1;
            }
            if (n.IndexOf("2") > -1 && DayOfWeek.Tuesday == current.DayOfWeek)
```

```
                {
                    workdays += -1;
                }
                if (n.IndexOf("3") > -1 == true && DayOfWeek.Wednesday ==
current.DayOfWeek)
                {
                    workdays += -1;
                }
                if (n.IndexOf("4") > -1 && DayOfWeek.Thursday == current.DayOfWeek)
                {
                    workdays += -1;
                }
                if (n.IndexOf("5") > -1 && DayOfWeek.Friday == current.DayOfWeek)
                {
                    workdays += -1;
                }
                if (n.IndexOf("6") > -1 && DayOfWeek.Saturday == current.DayOfWeek)
                {
                    workdays += -1;
                }
                if (n.IndexOf("7") > -1 && DayOfWeek.Sunday == current.DayOfWeek)
                {
                    workdays += -1;
                }
            }
            return workdays;
        }
        catch (Exception)
        {
            return 0;
        }
    }
```

```
private static int prior_sort()
{
    int b = 0;
    for (int a = 0; a < priority; a++)
    {
        if (Convert.ToDouble(TimeTable[a, 8]) >= Convert.ToDouble(TimeTable[b, 8]))
        {
            b = a;
        }
    }
    return b;
}
private bool data_sort(string dd1, string dd2)
{
    bool tt = false;
    DateTime startDate = DateTime.Parse(dd1);
    DateTime endDate = DateTime.Parse(dd2);
    if (startDate <= endDate)
    {
        tt = true;
    }
    return tt;
}
private static void prio_k()
{
    for (int a = 0; a < priority; a++)
    {
        int zzz = 0;
        double xxx = 0;
        string s2 = TimeTable[a, 4];
        string s1 = TimeTable[a, 3];
        zzz = workd(s1, s2) * Convert.ToInt32(Time[2]);
        xxx = Convert.ToDouble(TimeTable[a, 2]) / Convert.ToDouble(zzz);
```

```
            xxx = xxx + Convert.ToDouble(TimeTable[a, 1]);
            TimeTable[a, 8] = Convert.ToString(xxx);
        }
    }
    private static void testn()
    {
    }
    private static void grup() //групировка в расписание
    {
        int a = 0;
        int days = workd(Time[1], Time[0]);
        while (a < days)
        {
            for (int i = 0; i < Convert.ToInt32(Time[2]); i++)
            {
                if (TimeTable2[a, i] != null)
                {
                    string[] str2 = TimeTable2[a, i].Split('*');
                    if (str2[0] == "")
                    {
                        TimeTable2[a, i] = null;
                    }
                }
            }
            a++;
        };
        a = 0;
        while (a < days)
        {
            for (int i = 0; i < Convert.ToInt32(Time[2]) - 1; i++)
            {
                if (TimeTable2[a, i] != null)
                {
```

```
            int d = i + 1;

            string[] str1 = TimeTable2[a, i].Split('*');

            while (d < Convert.ToInt32(Time[2]))
            {
               if (TimeTable2[a, d] != null)
               {
                  string[] str2 = TimeTable2[a, d].Split('*');
                  if (str1[0] == str2[0])
                  {
                     string s = TimeTable2[a, i + 1];
                     TimeTable2[a, i + 1] = TimeTable2[a, d];
                     TimeTable2[a, d] = s;
                  }

               }
               d++;
            }
         }

      }
      a++;
   };
   a = 0;
   while (a < days)
   {
      for (int i = 0; i < Convert.ToInt32(Time[2]); i++)
      {
         if (TimeTable2[a, i] == null)
         {
            bool stop = false;
            int d = i + 1;
```

```
            while (stop == false)
            {
               if (d < Convert.ToInt32(Time[2]))
               {
                  if (TimeTable2[a, d] != null)
                  {
                     TimeTable2[a, i] = TimeTable2[a, d];
                     TimeTable2[a, d] = null;
                     stop = true;

                  }
               }
               else stop = true;
               d++;
            }
         }
      }
      a++;
};
a = 0;
while (a < days)
{
         int p = 1;
   for (int i = 0; i < Convert.ToInt32(Time[2]); i++)
   {
      for (int z = 0; z < Convert.ToInt32(Time[2]); z++)
      {
         if (TimeTable2[a, i] != null)
         {
            if (i + 1 < Convert.ToInt32(Time[2]))
            {
               if (TimeTable2[a, i + 1] != null)
               {
```

```
                    string[] str1 = TimeTable2[a, i].Split('*');
                    string[] str2 = TimeTable2[a, i + 1].Split('*');
                    if (str1[0] == str2[0])
                    {
                        p++;
                        TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
                        for (int n = i + 1; n < Convert.ToInt32(Time[2]); n++)
                        {
                            if (n < Convert.ToInt32(Time[2]) - 1)
                            {
                                TimeTable2[a, n] = TimeTable2[a, n + 1];
                            }
                            else
                            {
                                TimeTable2[a, n] = null;
                            }
                        }
                    }
                    else
                    {
                        bool tr = false;
                        try
                        {
                            TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*"
+str1[3];

                            tr = true;
                            p = 1;
                        }
                        catch (Exception)
                        {


                        }
```

```
                        if(tr==false)
                        {
                        p = 1;
                            TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
                        }
                    }
                }
                else
                {
                    string[] str1 = TimeTable2[a, i].Split('*');
                    TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
                }
            }
            else
            {
                string[] str1 = TimeTable2[a, i].Split('*');
                TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
            }

        }
    }


    }
    a++;
};
for (int b=0; b< days;b++)
{
    for(int c = 0; c < Convert.ToInt32(Time[2]); c++)
    {
```

```
                if (TimeTable2[b, c]!=null)
                {
                FullTT += TimeTable2[b, c] + "^";
                }
            }
            FullTT += "{";
        }
        string[] st = FullTT.Split('{');
            FullTT = null;
        for (int l=0;l<st.Length;l++)
        {
            if(st[l]!="")
            {
                FullTT += st[l] + "}";
            }
        }
        writeintt();
    }

    public static void writeintt() //запись в расписание
    {
        SqlConnection connectionTT = new SqlConnection(GetConnectionString());
        connectionTT.Open();
        SqlCommand command1 = new SqlCommand($"INSERT INTO Table2
(userID,UserTTs,DateOfCreate,DateStartTT,DateEndTT,WorkDay,Weekends)
VALUES('{Convert.ToInt16(Time[3])}',N'{FullTT}',N'{Convert.ToString(DateTime.Now)}',N'
{Convert.ToString(DateTime.Parse(Time[1]))}',N'{Convert.ToString(DateTime.Parse(Time[0]))
}','{Convert.ToInt16(Time[2])}',N'{n}')", connectionTT);
        command1.ExecuteNonQuery();
    }

    public static void SaveTT()
    {
```

```
            SqlConnection sqlConn1 = new SqlConnection(GetConnectionString());
            sqlConn1.Open();
            SqlConnection connectionTT = new SqlConnection(GetConnectionString());
            SqlCommand sqlCom1 = new SqlCommand($"SELECT * FROM Table2  WHERE
(UserId LIKE ('{Convert.ToInt16(Time[3])}'))", sqlConn1);
            SqlDataReader dr1 = sqlCom1.ExecuteReader();
            //результат запроса суем в таблицу
            DataTable dt1 = new DataTable();
            dt1.Load(dr1);
            int field = dt1.Rows[dt1.Rows.Count-1].Field<int>(0);
            idTT = field;
            SaveTTpart2();
        }
        public static void SaveTTpart2()
        {
            string str_tmp = input;
            string[] sub2 = str_tmp.Split('|');
            str_tmp = sub2[0];
            str_tmp = str_tmp.Remove(0, 1);
            str_tmp = str_tmp.Remove(str_tmp.Length - 2);
            //System.Windows.MessageBox.Show(str_tmp);
            string[] substrings1 = str_tmp.Split('^');
            int i = 0;
            for (int b = 0; b < substrings1.Length - 1; b++)
            {
                string[] str = substrings1[i].Split('*');
                for (int a = 0; a < 8; a++)
                {
                    Tasks[i, a] = str[a];
                }
                SqlConnection connectionTT = new SqlConnection(GetConnectionString());
                connectionTT.Open();
```

```
            SqlCommand command1 = new SqlCommand($"INSERT INTO Table3
(TaskName,TaskDeskription,TaskStartDate,TaskEndDate,TaskUserPriority,TaskLenth,TTid)
VALUES(N'{Tasks[i, 0]}',N'{Tasks[i, 5]}',N'{Convert.ToString(DateTime.Parse(Tasks[i,
3]))}',N'{Convert.ToString(DateTime.Parse(Tasks[i, 4]))}',N'{Tasks[i,
1]}','{Convert.ToInt16(Tasks[i, 2])}',N'{idTT}')", connectionTT);
                command1.ExecuteNonQuery();
                i++;
                //zadacha1*Високий*15*1.1.2015* 9.1.2015*й1й1й1й1*1*-6-7
            //    SqlConnection connectionTT1 = new SqlConnection(GetConnectionString());
            //connectionTT1.Open();
            //SqlCommand command1 = new SqlCommand($"INSERT INTO Table3
(TaskName,TaskDeskription,TaskStartDate,TaskEndDate,TaskUserPriority,TaskLenth,TTid)
VALUES(N'{Tasks[i, 0]}',N'{Tasks[i, 5]}',N'{Convert.ToString(DateTime.Parse(Tasks[i,
3]))}',N'{Convert.ToString(DateTime.Parse(Tasks[i, 4]))}',N'{Tasks[i,
1]}','{Convert.ToInt16(Tasks[i,2])}',N'{idTT}')", connectionTT1);
            //command1.ExecuteNonQuery();
            }


        }
        static private string GetConnectionString() //строка подключения
        {
            // To avoid storing the connection string in your code,
            // you can retrieve it from a configuration file.
            return "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\Database1.mdf;Integ
rated Security=True";
        }
    }
} namespace DiplomServer
{
    class workclass
    {
        static int port = 8005;
```

```
        static int priority = 0;
        static string input;
        private static string n;
        public static int idTT;
        public static string FullTT;
        private static string[] Time = new string[4];
        public static double pr;
        private static string[,] TimeTable = new string[1000, 10];
        private static string[,] Tasks = new string[1000, 10];
        static string[,] TimeTable2;

        public static void working()
        {
            input = "[zadacha1*Високий*15*1.1.2015* 9.1.2015*й1й1й1й1*1*-6-
7^zadacha2*Високий*10*2.1.2015* 9.1.2015*й2й2й2й2*1*-6-
7^zadacha3*Високий*20*1.1.2015* 9.1.2015*й3й3й3й3*1*-6-
7^10.1.2015*1.1.2015*10]^|timetable|2";
            parse();
            grup();
            SaveTT();
        }

        public static void logining()
        {
            input = "";

            IPEndPoint ipPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), port);

            // создаем сокет
            Socket listenSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            try
            {
```

```
            // связываем сокет с локальной точкой, по которой будем принимать данные
            listenSocket.Bind(ipPoint);

            // начинаем прослушивание
            listenSocket.Listen(10);

            while (true)
            {
        string ansv = "";
                Socket handler = listenSocket.Accept();
                // получаем сообщение
                StringBuilder builder = new StringBuilder();
                int bytes = 0; // количество полученных байтов
                byte[] data = new byte[2000000]; // буфер для получаемых данных

                do
                {
                   bytes = handler.Receive(data);
                   builder.Append(Encoding.UTF8.GetString(data, 0, bytes));
                }
                while (handler.Available > 0);
                // System.Windows.MessageBox.Show(DateTime.Now.ToShortTimeString() + ": "
+ builder.ToString());
                input = builder.ToString();


                //Table1TableAdapter a = new Table1TableAdapter();
                SqlConnection sqlConn = new SqlConnection(GetConnectionString());
                sqlConn.Open();
                //выполняем запрос

                string[] substrings2 = input.Split('^');
                string[] substrings1 = input.Split('^');
```

```
string[] substrings3= new string[10];
bool testTT = input.Contains("|");
if(testTT==true)
{
substrings3 = input.Split('|');
}
if (testTT==true)
{
   if(substrings3[1]== "timetable")
   {
   parse();
   grup();
   ansv = FullTT;
   SaveTT();
   }
}
else
{

   SqlCommand sqlCom = new SqlCommand($"SELECT * FROM Table1
WHERE (UserLogin LIKE (N'{substrings2[0]}'))", sqlConn);
   SqlDataReader dr = sqlCom.ExecuteReader();
   //результат запроса суем в таблицу
   DataTable dt = new DataTable();
   dt.Load(dr);
   //dt.Rows.Count;
   if (substrings2[2] != "reg")
   {
      bool ind = false;
      if (dt.Rows.Count != 0)
      {
         int id = -1;
         for (int i = 0; i < dt.Rows.Count; i++)
```

```
            {
                string field = dt.Rows[i].Field<string>(2);
                if (field == substrings2[1])
                {
                    id = i;
                    ind = true;
                }
            }
            if (ind == true)
            {

                ansv = "registred" + "^" + Convert.ToString(dt.Rows[id].Field<int>(0));
            }
            else
            {
                ansv = "vrong_pass";
            }

        }
        else
        {
            ansv = "vrong_login";

        }
    }
    else
    {
        //#region register
        //
        bool ind = false;
        if (dt.Rows.Count != 0)
        {
            ansv = "registred_yet";
```

```
                }
                else
                {
                    SqlConnection connection = new SqlConnection(GetConnectionString());
                        connection.Open();
                        SqlCommand command = new SqlCommand($"INSERT INTO Table1 (UserLogin,UserPass) VALUES(N'{substrings1[0]}',N'{substrings1[1]}')", connection);
                        command.ExecuteNonQuery();
                        ansv = "registred";

                    SqlConnection sqlConn1 = new SqlConnection(GetConnectionString());
                    sqlConn1.Open();
                    SqlCommand sqlCom1 = new SqlCommand($"SELECT * FROM Table1 WHERE (UserLogin LIKE (N'{substrings1[0]}'))", sqlConn1);
                    SqlDataReader redr = sqlCom1.ExecuteReader();
                    //результат запроса суем в таблицу
                    DataTable datb = new DataTable();
                    datb.Load(redr);
                    ansv += "^" + Convert.ToString(datb.Rows[0].Field<Int32>(0));
                }
              }
            }
            //SELECT* FROM[dbo].[student] WHERE(Surname LIKE(RTRIM(@Surname)+'%') OR Name LIKE(RTRIM(@Name)+'%') OR Fathername LIKE(@Fathername+'%'))

            // отправляем ответ
            //string message = "ваше сообщение доставлено";
            data = Encoding.UTF8.GetBytes(ansv);
            handler.Send(data);
            // закрываем сокет
            handler.Shutdown(SocketShutdown.Both);
            handler.Close();
```

```
            }
        }
        catch (Exception ex)
        {
            System.Windows.MessageBox.Show((ex.Message));
        }
    }

    public static void parse()
    {
        string str_tmp = input;
        string[] sub2 = str_tmp.Split('|');
        str_tmp = sub2[0];
        str_tmp = str_tmp.Remove(0, 1);
        str_tmp = str_tmp.Remove(str_tmp.Length - 2);
        //System.Windows.MessageBox.Show(str_tmp);
        string[] substrings1 = str_tmp.Split('^');
        int i = 0;
        for (int b = 0; b < substrings1.Length - 1; b++)
        {
            string[] str = substrings1[i].Split('*');
            for (int a = 0; a < 8; a++)
            {
                TimeTable[i, a] = str[a];
            }
            i++;
        }
        priority = substrings1.Length - 1;
        string[] str1 = substrings1[i].Split('*');
        string[] str2 = input.Split('|');
        Time[0] = str1[0];
        Time[1] = str1[1];
        Time[2] = str1[2];
```

```
            Time[3] = str2[2];
            Format();
        }

        public static void Format()
        {
            for (int a = 0; a < priority; a++)
            {
                if (TimeTable[a, 1] == "Низький")
                {
                    TimeTable[a, 1] = Convert.ToString(0.1);
                }
                else if (TimeTable[a, 1] == "Середній")
                {
                    TimeTable[a, 1] = Convert.ToString(0.25);
                }
                else
                {
                    TimeTable[a, 1] = Convert.ToString(0.5);
                }
            }

            try
            {
                n = TimeTable[0, 7];
                TimeTable2 = new string[workd(Time[1], Time[0]), Convert.ToInt32(Time[2])];
                //string[,] TimeTable2 = new string[10, 10];
                int days = Convert.ToInt32((DateTime.Parse(Time[0]) -
DateTime.Parse(Time[1])).Days + 1);
                DateTime startDate = DateTime.Parse(Time[1]);
                DateTime endDate = DateTime.Parse(Time[0]);
                DateTime current = startDate;
                int workdays = days;
```

```
int t = 0;
for (int i = 0; i < days; i++)
{
   t = 0;
   current = startDate.AddDays(i);
   prio_k();
   testn();
   if (n.IndexOf("1") == -1 && DayOfWeek.Monday == current.DayOfWeek)
   {
      do
      {
         int r = 0;
         prio_k();
         r = prior_sort();
         if (TimeTable[r, 0] != "")
         {
            TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
            TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
         }
         if (TimeTable[r, 2] == "0")
         {
            TimeTable[r, 0] = null;
            TimeTable[r, 1] = null;
            TimeTable[r, 2] = null;
            TimeTable[r, 3] = null;
            TimeTable[r, 4] = null;
            TimeTable[r, 5] = null;
            TimeTable[r, 6] = null;
            TimeTable[r, 7] = null;
            TimeTable[r, 8] = null;
         }
         prior_sort();
```

```
                t++;
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
        }
        testn();
        if (n.IndexOf("2") == -1 && DayOfWeek.Tuesday == current.DayOfWeek)
        {
            do
            {
                int r = 0;
                prio_k();
                r = prior_sort();
                if (TimeTable[r, 0] != "")
                {
                    TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                    TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                }
                if (TimeTable[r, 2] == "0")
                {
                    TimeTable[r, 0] = null;
                    TimeTable[r, 1] = null;
                    TimeTable[r, 2] = null;
                    TimeTable[r, 3] = null;
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
                    TimeTable[r, 8] = null;
                }
                prior_sort();
                t++;
            } while (t < Convert.ToInt32(Time[2]));
```

```
            t = 0;
        }
        testn();
        if (n.IndexOf("3") == -1 && DayOfWeek.Wednesday == current.DayOfWeek)
        {
            do
            {
                int r = 0;
                prio_k();
                r = prior_sort();
                if (TimeTable[r, 0] != "")
                {
                    TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                    TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                }
                if (TimeTable[r, 2] == "0")
                {
                    TimeTable[r, 0] = null;
                    TimeTable[r, 1] = null;
                    TimeTable[r, 2] = null;
                    TimeTable[r, 3] = null;
                    TimeTable[r, 4] = null;
                    TimeTable[r, 5] = null;
                    TimeTable[r, 6] = null;
                    TimeTable[r, 7] = null;
                    TimeTable[r, 8] = null;
                }
                prior_sort();
                t++;
            } while (t < Convert.ToInt32(Time[2]));
            t = 0;
        }
```

```
            testn();
            if (n.IndexOf("4") == -1 && DayOfWeek.Thursday == current.DayOfWeek)
            {
               do
               {
                  int r = 0;
                  prio_k();
                  r = prior_sort();
                  if (TimeTable[r, 0] != "")
                  {
                     TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                     TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                  }
                  if (TimeTable[r, 2] == "0")
                  {
                     TimeTable[r, 0] = null;
                     TimeTable[r, 1] = null;
                     TimeTable[r, 2] = null;
                     TimeTable[r, 3] = null;
                     TimeTable[r, 4] = null;
                     TimeTable[r, 5] = null;
                     TimeTable[r, 6] = null;
                     TimeTable[r, 7] = null;
                     TimeTable[r, 8] = null;
                  }
                  prior_sort();
                  t++;
               } while (t < Convert.ToInt32(Time[2]));
               t = 0;
            }
            testn();
            if (n.IndexOf("5") == -1 && DayOfWeek.Friday == current.DayOfWeek)
```

```
            {
                do
                {
                    int r = 0;
                    prio_k();
                    r = prior_sort();
                    if (TimeTable[r, 0] != "")
                    {
                        TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                        TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                    }
                    if (TimeTable[r, 2] == "0")
                    {
                        TimeTable[r, 0] = null;
                        TimeTable[r, 1] = null;
                        TimeTable[r, 2] = null;
                        TimeTable[r, 3] = null;
                        TimeTable[r, 4] = null;
                        TimeTable[r, 5] = null;
                        TimeTable[r, 6] = null;
                        TimeTable[r, 7] = null;
                        TimeTable[r, 8] = null;
                    }
                    prior_sort();
                    t++;
                } while (t < Convert.ToInt32(Time[2]));
                t = 0;
            }
            testn();
            if (n.IndexOf("6") == -1 && DayOfWeek.Saturday == current.DayOfWeek)
            {
                do
```

```
                {
                    int r = 0;
                    prio_k();
                    r = prior_sort();
                    if (TimeTable[r, 0] != "")
                    {
                        TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                        TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                    }
                    if (TimeTable[r, 2] == "0")
                    {
                        TimeTable[r, 0] = null;
                        TimeTable[r, 1] = null;
                        TimeTable[r, 2] = null;
                        TimeTable[r, 3] = null;
                        TimeTable[r, 4] = null;
                        TimeTable[r, 5] = null;
                        TimeTable[r, 6] = null;
                        TimeTable[r, 7] = null;
                        TimeTable[r, 8] = null;
                    }
                    prior_sort();
                    t++;
                } while (t < Convert.ToInt32(Time[2]));
                t = 0;
            }
            testn();
            if (n.IndexOf("7") == -1 && DayOfWeek.Sunday == current.DayOfWeek)
            {
                do
                {
                    int r = 0;
```

```
                    prio_k();
                    r = prior_sort();
                    if (TimeTable[r, 0] != "")
                    {
                       TimeTable2[i, t] = TimeTable[r, 0] + "*" + TimeTable[r, 5] + "*" +
Convert.ToString(current);
                       TimeTable[r, 2] = Convert.ToString(Convert.ToInt32(TimeTable[r, 2]) - 1);
                    }

                    if (TimeTable[r, 2] == "0")
                    {
                       TimeTable[r, 0] = null;
                       TimeTable[r, 1] = null;
                       TimeTable[r, 2] = null;
                       TimeTable[r, 3] = null;
                       TimeTable[r, 4] = null;
                       TimeTable[r, 5] = null;
                       TimeTable[r, 6] = null;
                       TimeTable[r, 7] = null;
                       TimeTable[r, 8] = null;
                    }
                    prior_sort();
                    t++;
                 } while (t < Convert.ToInt32(Time[2]));
                 t = 0;
              }



           }
        }
        catch (Exception)
        {
```

```
        }


    }



    private static int workd(string dd1, string dd2)
    {
        try
        {
            DateTime startDate = DateTime.Parse(dd1);
            DateTime endDate = DateTime.Parse(dd2);
            DateTime current = startDate;
            int days = (endDate - startDate).Days + 1;
            int workdays = days;
            for (int i = 0; i < days; i++)
            {
                testn();
                current = startDate.AddDays(i);
                if (n.IndexOf("1") > -1 && DayOfWeek.Monday == current.DayOfWeek)
                {
                    workdays += -1;
                }
                if (n.IndexOf("2") > -1 && DayOfWeek.Tuesday == current.DayOfWeek)
                {
                    workdays += -1;
                }
                if (n.IndexOf("3") > -1 == true && DayOfWeek.Wednesday ==
current.DayOfWeek)
                {
                    workdays += -1;
```

```
            }
            if (n.IndexOf("4") > -1 && DayOfWeek.Thursday == current.DayOfWeek)
            {
               workdays += -1;
            }
            if (n.IndexOf("5") > -1 && DayOfWeek.Friday == current.DayOfWeek)
            {
               workdays += -1;
            }
            if (n.IndexOf("6") > -1 && DayOfWeek.Saturday == current.DayOfWeek)
            {
               workdays += -1;
            }
            if (n.IndexOf("7") > -1 && DayOfWeek.Sunday == current.DayOfWeek)
            {
               workdays += -1;
            }
         }
         return workdays;
      }
      catch (Exception)
      {
         return 0;
      }
   }

   private static int prior_sort()
   {
      int b = 0;
      for (int a = 0; a < priority; a++)
      {
         if (Convert.ToDouble(TimeTable[a, 8]) >= Convert.ToDouble(TimeTable[b, 8]))
         {
```

```
            b = a;
        }
    }
    return b;
}
private bool data_sort(string dd1, string dd2)
{
    bool tt = false;
    DateTime startDate = DateTime.Parse(dd1);
    DateTime endDate = DateTime.Parse(dd2);
    if (startDate <= endDate)
    {
        tt = true;
    }
    return tt;
}
private static void prio_k()
{
    for (int a = 0; a < priority; a++)
    {
        int zzz = 0;
        double xxx = 0;
        string s2 = TimeTable[a, 4];
        string s1 = TimeTable[a, 3];
        zzz = workd(s1, s2) * Convert.ToInt32(Time[2]);
        xxx = Convert.ToDouble(TimeTable[a, 2]) / Convert.ToDouble(zzz);
        xxx = xxx + Convert.ToDouble(TimeTable[a, 1]);
        TimeTable[a, 8] = Convert.ToString(xxx);
    }
}
private static void testn()
{
}
```

```
private static void grup() //групировка в расписание
{
    int a = 0;
    int days = workd(Time[1], Time[0]);
    while (a < days)
    {
        for (int i = 0; i < Convert.ToInt32(Time[2]); i++)
        {
            if (TimeTable2[a, i] != null)
            {
                string[] str2 = TimeTable2[a, i].Split('*');
                if (str2[0] == "")
                {
                    TimeTable2[a, i] = null;
                }
            }
        }
        a++;
    };
    a = 0;
    while (a < days)
    {
        for (int i = 0; i < Convert.ToInt32(Time[2]) - 1; i++)
        {
            if (TimeTable2[a, i] != null)
            {
                int d = i + 1;

                string[] str1 = TimeTable2[a, i].Split('*');

                while (d < Convert.ToInt32(Time[2]))
                {
                    if (TimeTable2[a, d] != null)
```

```
                {
                    string[] str2 = TimeTable2[a, d].Split('*');
                    if (str1[0] == str2[0])
                    {
                        string s = TimeTable2[a, i + 1];
                        TimeTable2[a, i + 1] = TimeTable2[a, d];
                        TimeTable2[a, d] = s;
                    }

                }
                d++;
            }
        }

    }
    a++;
};
a = 0;
while (a < days)
{
    for (int i = 0; i < Convert.ToInt32(Time[2]); i++)
    {
        if (TimeTable2[a, i] == null)
        {
            bool stop = false;
            int d = i + 1;
            while (stop == false)
            {
                if (d < Convert.ToInt32(Time[2]))
                {
                    if (TimeTable2[a, d] != null)
                    {
                        TimeTable2[a, i] = TimeTable2[a, d];
```

```
                        TimeTable2[a, d] = null;
                        stop = true;


                    }
                }
                else stop = true;
                d++;
            }
        }
    }
    a++;
};
a = 0;
while (a < days)
{
        int p = 1;
    for (int i = 0; i < Convert.ToInt32(Time[2]); i++)
    {
        for (int z = 0; z < Convert.ToInt32(Time[2]); z++)
        {
            if (TimeTable2[a, i] != null)
            {
                if (i + 1 < Convert.ToInt32(Time[2]))
                {
                    if (TimeTable2[a, i + 1] != null)
                    {
                        string[] str1 = TimeTable2[a, i].Split('*');
                        string[] str2 = TimeTable2[a, i + 1].Split('*');
                        if (str1[0] == str2[0])
                        {
                            p++;
                            TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
```

```
                    for (int n = i + 1; n < Convert.ToInt32(Time[2]); n++)
                    {
                        if (n < Convert.ToInt32(Time[2]) - 1)
                        {
                            TimeTable2[a, n] = TimeTable2[a, n + 1];
                        }
                        else
                        {
                            TimeTable2[a, n] = null;
                        }
                    }
                }
                else
                {
                    bool tr = false;
                    try
                    {
                        TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*"
+str1[3];
                        tr = true;
                        p = 1;
                    }
                    catch (Exception)
                    {

                    }
                    if(tr==false)
                    {
                    p = 1;
                        TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
                    }
                }
```

```
                    }
                    else
                    {
                        string[] str1 = TimeTable2[a, i].Split('*');
                        TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
                    }
                }
                else
                {
                    string[] str1 = TimeTable2[a, i].Split('*');
                    TimeTable2[a, i] = str1[0] + "*" + str1[1] + "*" + str1[2] + "*" +
Convert.ToString(p);
                }

            }
        }


    }
    a++;
};
for (int b=0; b< days;b++)
{
    for(int c = 0; c < Convert.ToInt32(Time[2]); c++)
    {
        if (TimeTable2[b, c]!=null)
        {
        FullTT += TimeTable2[b, c] + "^";
        }
    }
    FullTT += "{";
}
```

```
            string[] st = FullTT.Split('{');
                FullTT = null;
            for (int l=0;l<st.Length;l++)
            {
                if(st[l]!="")
                {
                    FullTT += st[l] + "}";
                }
            }
            writeintt();
        }

        public static void writeintt() //запись в расписание
        {
            SqlConnection connectionTT = new SqlConnection(GetConnectionString());
            connectionTT.Open();
            SqlCommand command1 = new SqlCommand($"INSERT INTO Table2
(userID,UserTTs,DateOfCreate,DateStartTT,DateEndTT,WorkDay,Weekends)
VALUES('{Convert.ToInt16(Time[3])}',N'{FullTT}',N'{Convert.ToString(DateTime.Now)}',N'
{Convert.ToString(DateTime.Parse(Time[1]))}',N'{Convert.ToString(DateTime.Parse(Time[0]))
}','{Convert.ToInt16(Time[2])}',N'{n}')", connectionTT);
            command1.ExecuteNonQuery();
        }

        public static void SaveTT()
        {
            SqlConnection sqlConn1 = new SqlConnection(GetConnectionString());
            sqlConn1.Open();
            SqlConnection connectionTT = new SqlConnection(GetConnectionString());
            SqlCommand sqlCom1 = new SqlCommand($"SELECT * FROM Table2  WHERE
(UserId LIKE ('{Convert.ToInt16(Time[3])}'))", sqlConn1);
            SqlDataReader dr1 = sqlCom1.ExecuteReader();
            //результат запроса суем в таблицу
```

```
            DataTable dt1 = new DataTable();
            dt1.Load(dr1);
            int field = dt1.Rows[dt1.Rows.Count-1].Field<int>(0);
            idTT = field;
            SaveTTpart2();
        }
        public static void SaveTTpart2()
        {
            string str_tmp = input;
            string[] sub2 = str_tmp.Split('|');
            str_tmp = sub2[0];
            str_tmp = str_tmp.Remove(0, 1);
            str_tmp = str_tmp.Remove(str_tmp.Length - 2);
            //System.Windows.MessageBox.Show(str_tmp);
            string[] substrings1 = str_tmp.Split('^');
            int i = 0;
            for (int b = 0; b < substrings1.Length - 1; b++)
            {
                string[] str = substrings1[i].Split('*');
                for (int a = 0; a < 8; a++)
                {
                    Tasks[i, a] = str[a];
                }
                SqlConnection connectionTT = new SqlConnection(GetConnectionString());
                connectionTT.Open();
                SqlCommand command1 = new SqlCommand($"INSERT INTO Table3
(TaskName,TaskDeskription,TaskStartDate,TaskEndDate,TaskUserPriority,TaskLenth,TTid)
VALUES(N'{Tasks[i, 0]}',N'{Tasks[i, 5]}',N'{Convert.ToString(DateTime.Parse(Tasks[i,
3]))}',N'{Convert.ToString(DateTime.Parse(Tasks[i, 4]))}',N'{Tasks[i,
1]}','{Convert.ToInt16(Tasks[i, 2])}',N'{idTT}')", connectionTT);
                command1.ExecuteNonQuery();
                i++;
                //zadacha1*Високий*15*1.1.2015* 9.1.2015*й1й1й1й1*1*-6-7
```

```
        //    SqlConnection connectionTT1 = new SqlConnection(GetConnectionString());
        //connectionTT1.Open();
        //SqlCommand command1 = new SqlCommand($"INSERT INTO Table3
(TaskName,TaskDeskription,TaskStartDate,TaskEndDate,TaskUserPriority,TaskLenth,TTid)
VALUES(N'{Tasks[i, 0]}',N'{Tasks[i, 5]}',N'{Convert.ToString(DateTime.Parse(Tasks[i,
3]))}',N'{Convert.ToString(DateTime.Parse(Tasks[i, 4]))}',N'{Tasks[i,
1]}','{Convert.ToInt16(Tasks[i,2])}',N'{idTT}')", connectionTT1);
        //command1.ExecuteNonQuery();
        }


    }
    static private string GetConnectionString() //строка подключения
    {
        // To avoid storing the connection string in your code,
        // you can retrieve it from a configuration file.
        return "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\\Database1.mdf;Integ
rated Security=True";
    }
  }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**В.о. завідувача кафедри**
_____ *С.Л. Проскура*	_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“16” квітня 2018 р.	“17” квітня 2018 р.

Клієнт-серверна аналітична система управління робочим часом

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5213.1181-с.ТЗ*

на 9 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: *клієнт-серверна аналітична програма управління робочим часом*

1.2 Найменування організації-замовника та організації-учасника робіт

Генеральним замовником проекту являється кафедра Автоматизованих систем обробки інформації та управління НТУУ "КПІ". Представником замовника є Проскура Світлана Леонідівна.

Розробником системи є студент групи ІС-52 факультету інформатики та обчислювальної техніки НТУУ «КПІ ім. Ігоря Сікорського» Косяк Олександр Миколайович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням автоматизованого пошуку оптимального шляху робота-рятівника – 5 лютого 2018 рік.

Плановий термін по закінченню роботи над створенням автоматизованого пошуку оптимального шляху робота-рятівника – не пізніше 1 червня 2019 року.

2 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ КОМПЛЕКСУ ЗАДАЧ

2.1 Призначення комплексу задач

Призначенням розробки є створення клієнт-серверної аналітичної системи управління робочим часом.

2.2 Цілі створення комплексу задач

Цілями розробки є спрощення процесу створення розкладу робочого часу користувача, для чого потрібно:

- знайти та проаналізувати аналоги на предмет ефективності роботи;
- вирішити задачу автоматизації процесу генерації розпорядку;
- зібрати інформацію про ефективність різних підходів до використання робочого часу;
- створити ПЗ, що дозволить багатьом користувачам використовувати потужності серверу, а не навантажувати власний компьютер;

Для досягнення поставлених цілей необхідно вирішити наступні задачі:

- отримати дані від користувача;
- провести аналіз отриманих даних на сервері;
- по проведеному аналізу згенерувати кілька варіантів попереднього розкладу;
- надати користувачу можливість скорегувати згенеровані варіанти розкладу в разі необхідності.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Працювати з застосуванням можуть всі користувачі Windows. З його допомогою користувач може отримати готовий розклад що буде згенеровано на основі його даних, та матиме змогу відредагувати кінцевий варіант в разі потреби.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Дане розширення для Windows має генерувати розклад згідно заданих налаштувань та виконувати наступні функції:

- система повинна надати користувачу можливість вводити дані задач;
- система повинна надавати користувачу можливість змінювати окремі результати згенерованого розкладу, без необхідності заново генерувати весь розклад.

4.2 Вимоги до надійності

Система повинна зберігати працездатність і забезпечувати відновлення своїх функцій при виникненні наступних позаштатних ситуацій:

- Задання надто великої генерації, яку не може обробити машина;

Система повинна бути працездатною після перезапуску заштатних ситуацій.

4.3 Вимоги до складу і параметрів технічних засобів

Склад, структура і способи організації даних в системі повинні бути визначені на етапі технічного проектування.

Структура технічних засобів визначається виходячи із можливості їх забезпечити процедуру генерації міста вказаного розміру та складності.

Для правильної роботи розробленої системи до складу технічних засобів повинен входити комп’ютер, що має конфігурацію наведену нижче:

- комп’ютер з такою конфігурацією:

1) двох ядерний процесор з частотою не нижче 2.5 ГГц;

2) достатній об’єм оперативної пам’яті (не менше 4 ГБ);

3) відеокарта NVIDIA GeForce 470 GTX або AMD Radeon 6870 HD series.

- додатково має бути встановлене таке програмне забезпечення:

    1) операційна система Windows 10 64-bit;

    2) Visual Studio 2017 або новіша;

- комп’ютерна периферія, до складу якої входить:

    1) монітор;

    2) мишка або тачпад;

    3) клавіатура.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з автоматизації формування асортименту торгівельної організації.

№ п/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1.	Підготовка технічного завдання на розробку програмного продукту	07.02.2019	
2.	Розробка сценарію роботи	12.02.2019	
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	20.02.2019	
4.	Узгодження з керівником інтерфейсу користувача	02.03.2019	
5.	Розробка інформаційного забезпечення	17.03.2019	
6.	Розробка програмного забезпечення	29.03.2019	
7.	Налагодження програми	27.04.2019	
8.	Тестування програми	30.05.2019	
9.	Здача готового програмного продукту замовнику	12.05.2019	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Для контролю правильності роботи програмного забезпечення буде проведено модульне тестування (Unit Test). В ході тестування будуть перевірені всі граничні умови вхідних даних, будуть перевірені всі вищеобумовлені вимоги. Буде проведено випробування коректності роботи алгоритмів при автоматичній генерації даних та отримання даних з файлу.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**В.о. завідувача кафедри**
_____ *С.Л. Проскура*	_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“16” травня 2019 р.	“17” травня 2019 р.

Клієнт-серверна аналітична система управління робочим часом

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5213.1181-с.ПМВ*

на 10 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Повне найменування системи: клієнт-серверна аналітична система управління робочим часом.

1.2 Область застосування

Застосунок для клієнтів, що мають потреби в управлінні робочим часом.

1.3 Умовне позначення програми

Скорочене найменування системи: Timetable Generator.

2 МЕТА ВИПРОБУВАНЬ

Метою випробувань являється перевірка відповідності функцій клієнт-серверної аналітичної системи управління робочим часом вимогам технічного завдання.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до структури та функціонування Системи

Timetable Generator включає наступні підсистеми:

- перевірка внесеної інформації;
- генерування розкладу;
- генерація файлу розкладу.

3.2 Вимоги до надійності системи та збереженості інформації

Аварійні ситуації, які мають враховуватись при розробці Системи - наступні:

- невірне введення даних користувачем;
- контроль вводу даних;
- збій електроживлення;
- збій у роботі загальносистемного програмного забезпечення;

Запобігання невірному введенню даних користувачем реалізовано на стадії введення даних розкладу у все існуючому програмному забезпеченні.

Система повинна бути стійка до збоїв у апаратному забезпеченні, відмови у роботі обладнання або зникнення напруги. Така стійкість може бути забезпечена за рахунок використання систем резервного копіювання та використання джерел безперебійного живлення.

Має бути забезпечено збереженість інформації або можливість її відновлення при вимкненнях живлення електромережі, неправильних діях персоналу, „вірусних атаках” та відмовах компонентів технічного забезпечення.

3.3 Вимоги до програмного забезпечення

Так як даний продукт являє собою клієнт-серверну систему, користувачам не потрібно встановлювати все програмне забезпечення. Встановлений клієнтський додаток необхідно розмістити на окремий комп'ютер в якому необхідні такі мінімальні характеристики:

- процесор – Intel Core i3 5покоління і вище;
- об’єм оперативної пам’яті – 2 Gb і більше;
- дискретна відеокарта з пам’ятю 256 Mb і більше;
- швидкий доступ до бази даних через мережу Інтернет;
- інші характеристики незначним чином впливають на роботу.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Випробування проводяться на основі наступних документів:

- ГОСТ 34.60̃92. Інформаційна технологія. Види випробувань автоматизованих систем;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Порядок проведення випробувань наступний:

Запуск програми відбувається з запуску клієнтського додатку. Саме з цього етапу починається випробування.

6 РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

В процесі тестування були перевірена уся функціональність комплексу задач. У наступних таблицях наведений перелік випробувань основних функціональних можливостей (таблиця 6.1 – таблиця 6.6)

Таблиця 6.1 – Перевірка підключення до сервера зі сторони клієнта

Мета тесту:	Перевірка функції «Підключення(Клієнт)»
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі.
Очікуваний результат:	Клієнтська програма запущена; При введенні логіну та паролю з’являється відповідне повідомлення
Вхідні дані:	Логін та пароль
Схема проведення тесту:	Після запуску клієнтського додатку ввести інформацію в поля логін та пароль; Натиснути кнопку «Зареєструватись»;

Таблиця 6.2 – Перевірка відправки інформації клієнтом

Мета тесту:	Перевірка відправки інформації клієнтом
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Набір певних задач.
Схема проведення тесту:	Ввести в програмі клієнта набір задач, та натиснути кнопку згенерувати розклад.
Очікуваний результат:	На серверній програмі, в панелі інформації з’явиться інформація про отримання від клієнту початкових даних для генерації розпорядку

Таблиця 6.3 – Перевірка планування зі сторони сервера

Мета тесту:	Перевірка функції «Планування(сервер)»
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Інформація з клієнту
Схема проведення тесту:	Здійснити тест описаний у 6.2;
Очікуваний результат:	На серверній програмі, в панелі інформації з’явиться інформація про те що згенеровано новий розклад для клієнта, при натисненні на кнопку «деталі», можна буде його переглянути

Таблиця 6.4 – Перевірка відправки зі сторони сервер

Мета тесту:	Перевірка функції «Відправка (Сервер)»
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Виконати дії з тесту 6.2.
Схема проведення тесту:	Дочекатись надходження повідомлення від серверу з згенерованим розкладом;
Очікуваний результат:	З’явиться відповідне повідомлення, про те що ваш розклад згенеровано

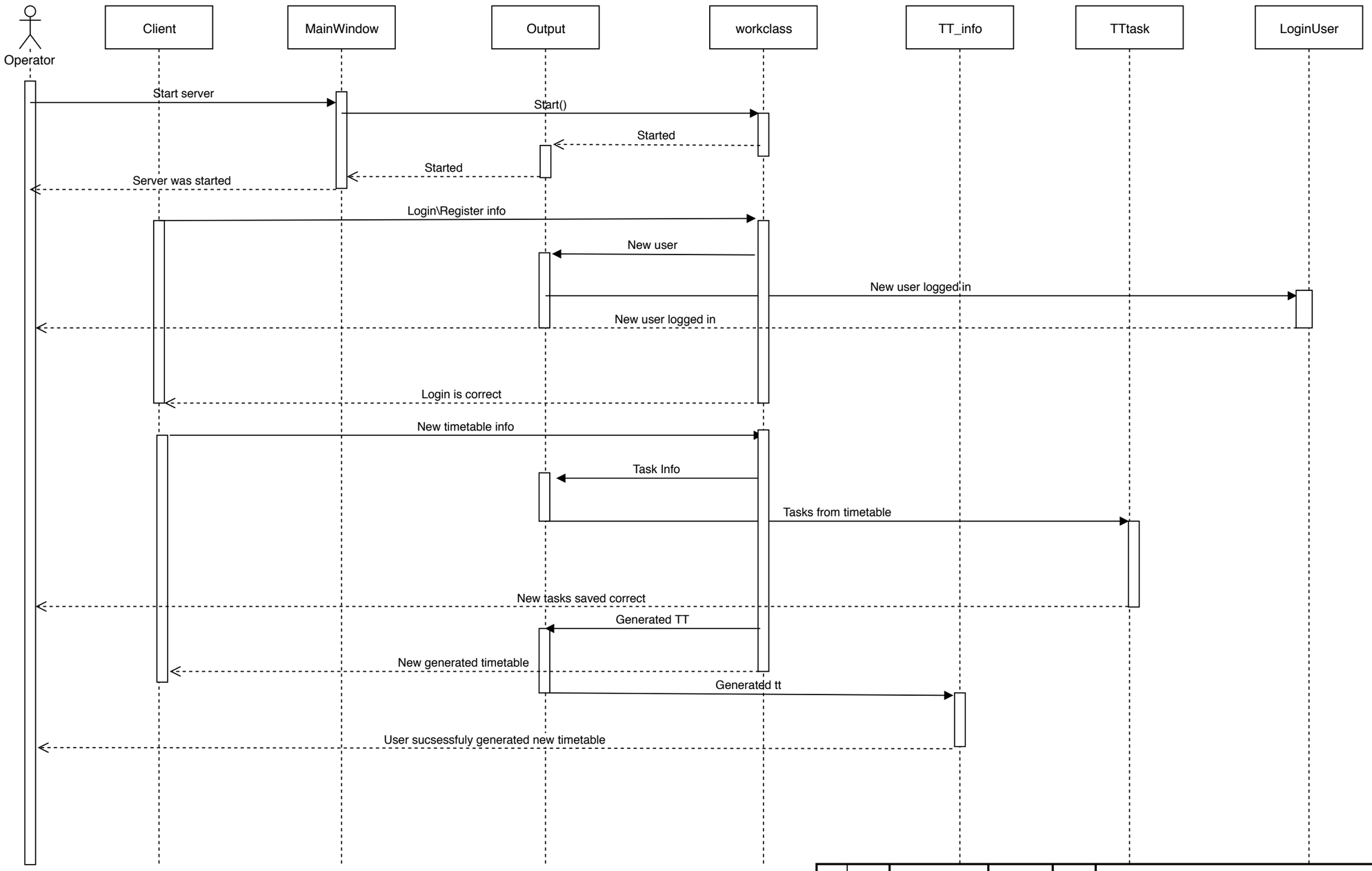
Таблиця 6.5 – Перевірка функції «Convert» (клієнт-сервер)

Мета тесту:	Перевірка функції «Convert» (клієнт-сервер)
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Інформація з клієнта.
Схема проведення тесту:	Виконати дії тесту 6.2 Дочекатись надходження інформації від клієнта;
Очікуваний результат:	При перегляді вхідних даних на сервер немає різниці між інформацією введеною в клієнт

Таблиця 6.6 – Перевірка функції «Convert» (сервер клієнт)

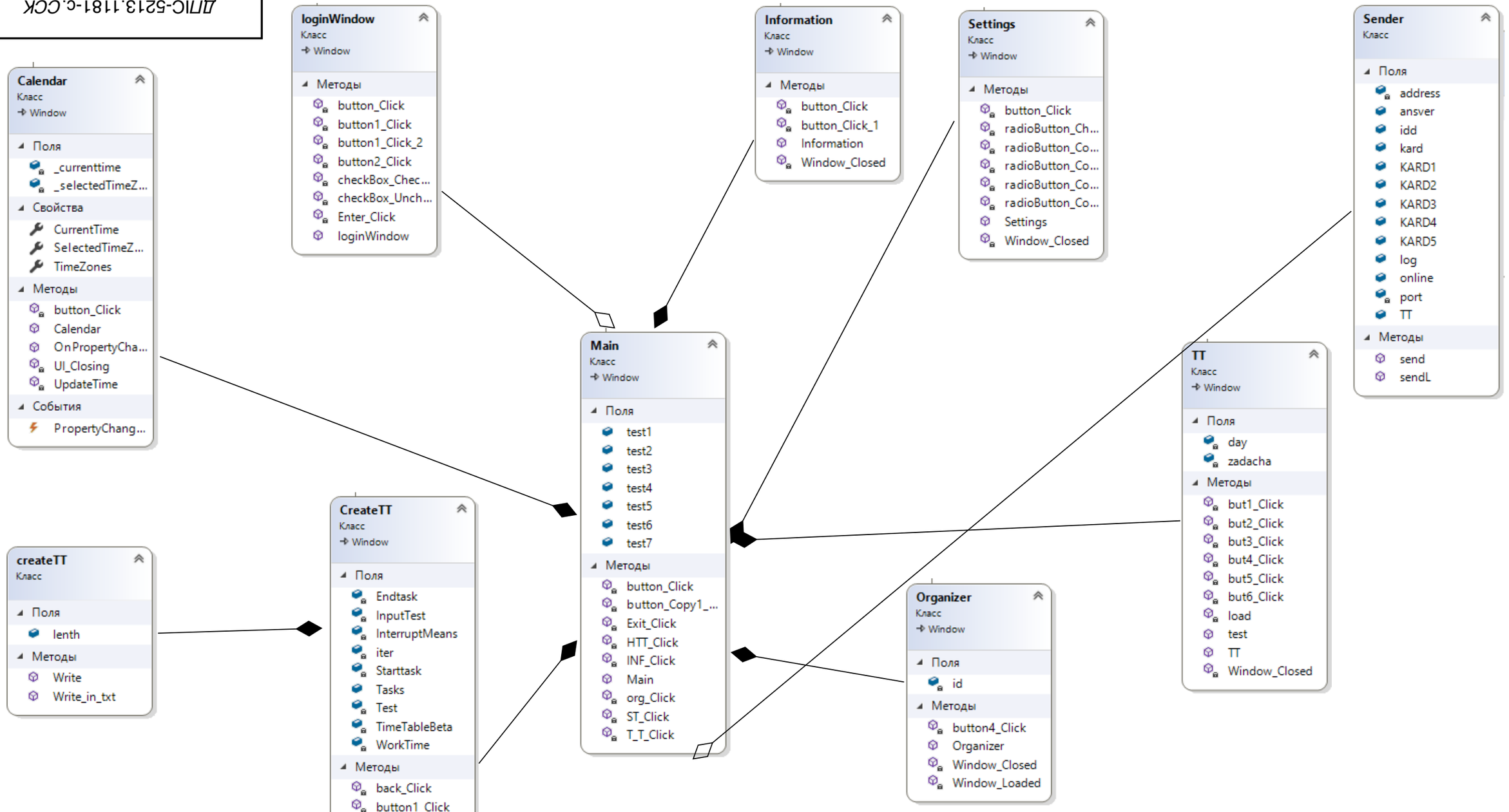
Мета тесту:	Перевірка функції «Convert» (сервер клієнт)
Початковий стан:	Комп’ютер увімкнено; ОС запущена та працює у стаціонарному режимі; Клієнтська програма запущена і працює у штатному режимі; Серверна програма запущена і працює у штатному режимі;
Вхідні дані:	Виконати дії тесту 6.2
Схема проведення тесту:	Дочекатись надходження інформації з серверу.
Очікуваний результат:	Розклад що було згенеровано на сервері не відрізняється від розкладу що відображається на клієнті

ДПІС-5213.1181-с.ССП

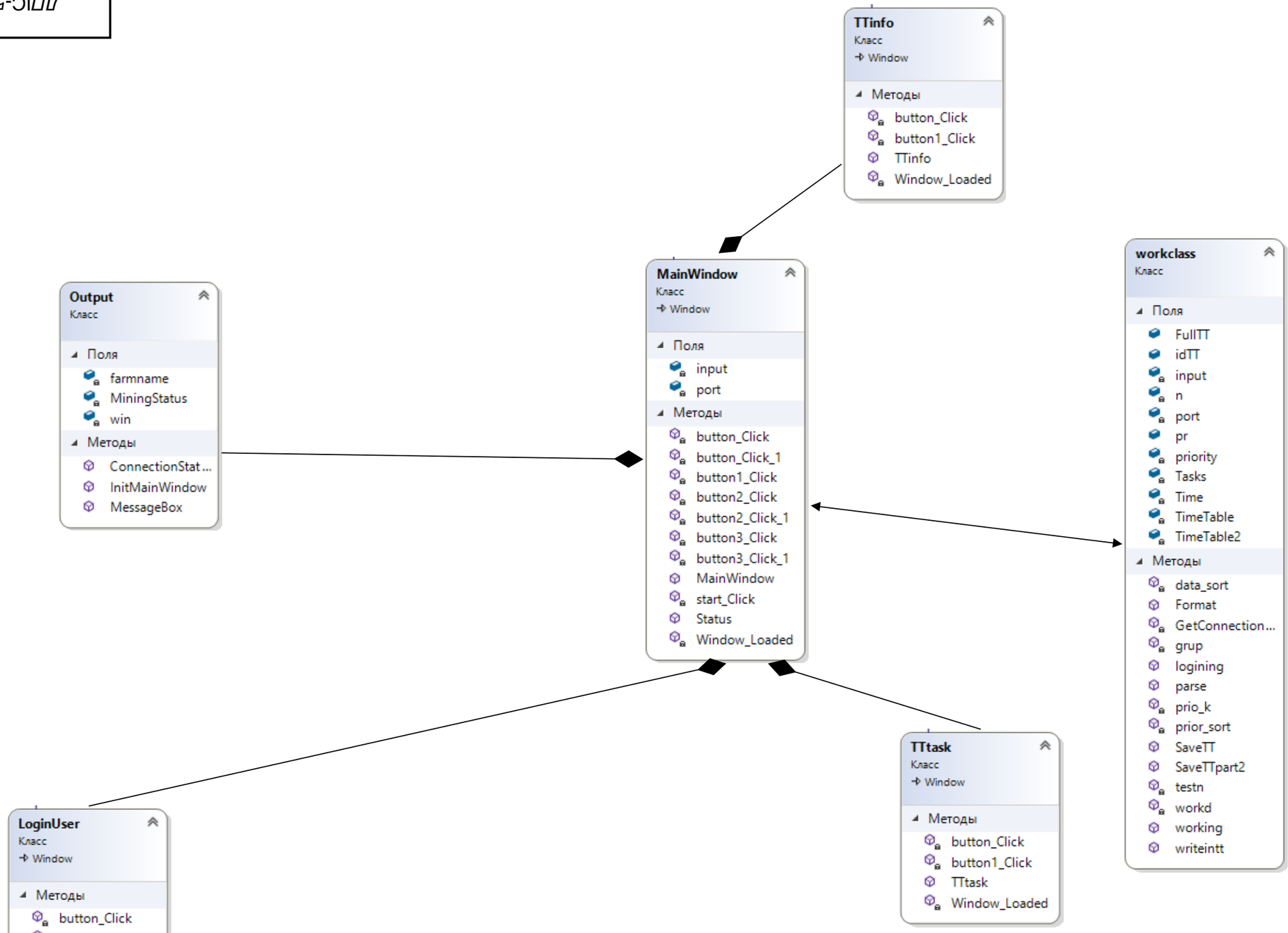


					ДПІС-5213.1181-с.*ССП*	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна послідовності*	
Розробив		Косяк О.М.				Аркуш 1 / Аркушів 1
Перевірив		Проскура С.Л.				
Н. кон.		Халус О. А.			Клієнт-серверна аналітична система управління робочим часом	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*
Затвердив		Павлов О.А.				

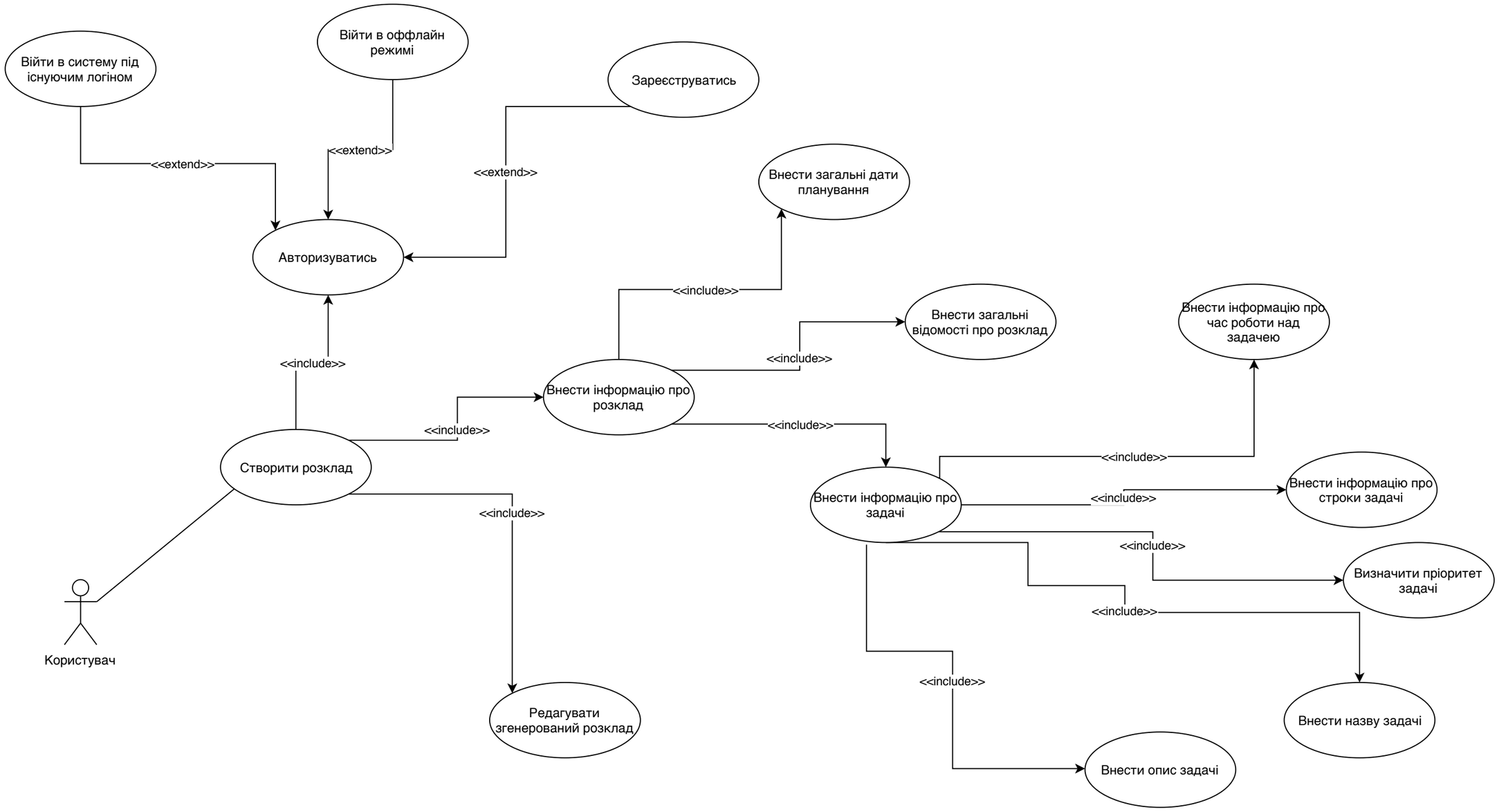
ДПІС-5213.1181-с.ССК



					ДПІС-5213.1181-с.*ССК*	
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна класів*	Літера / Маса / Масштаб
Розробив		Косяк О.М.				
Перевірив		Проскура С.Л.				Аркуш 1 / Аркушів 1
Н. кон.		Халус О. А.			Клієнт-серверна аналітична система управління робочим часом	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*
Затвердив		Павлов О.А.				

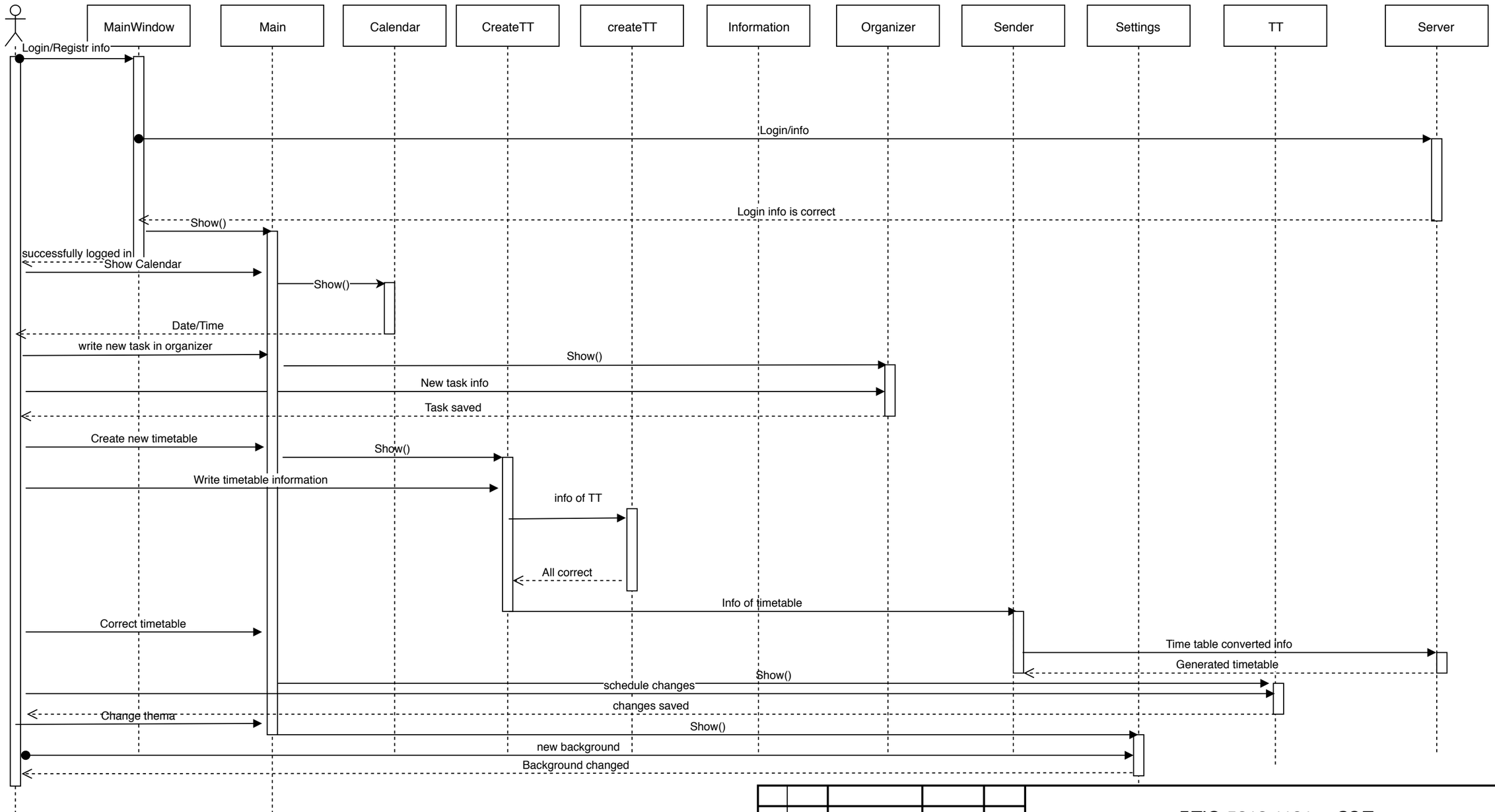


					ДПІС-5213.1181-с.*ССК*	
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна класів*	Літера / Маса / Масштаб
Розробив		Косяк О.М.				
Перевірив		Проскура С.Л.				Аркуш 1 / Аркушів 1
Н. кон.		Халус О. А.			Клієнт-серверна аналітична система управління робочим часом	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*
Затвердив		Павлов О.А.				



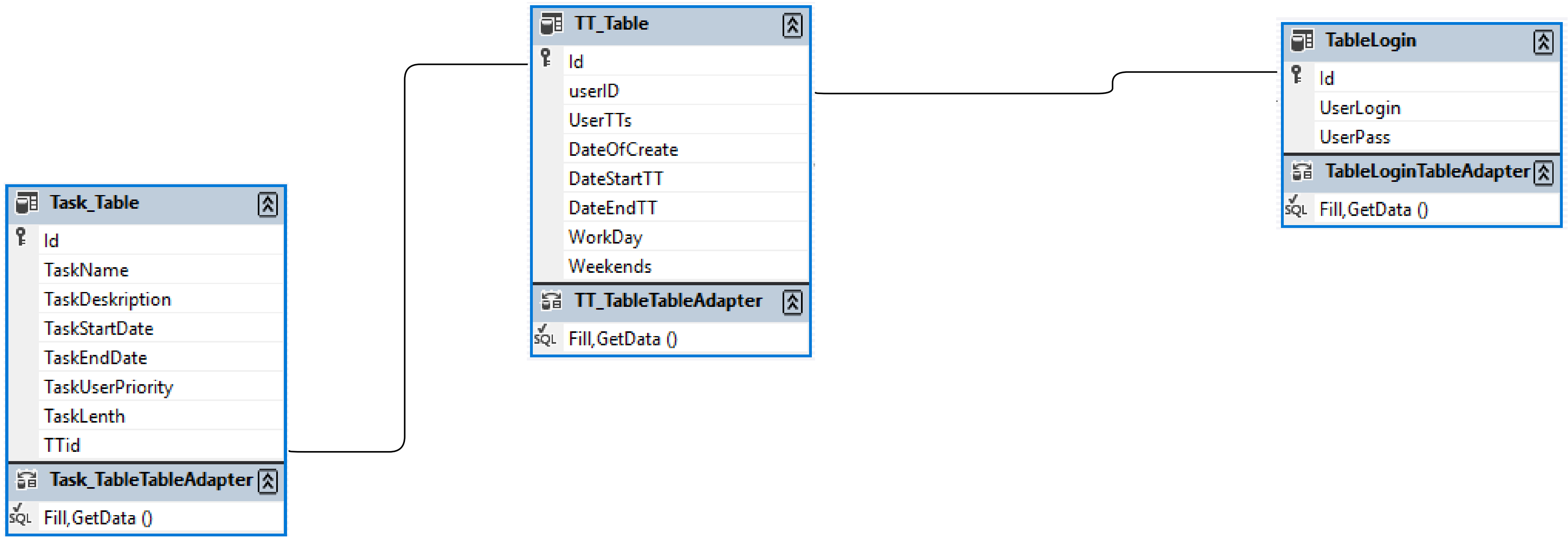
					ДПІС-5213.1181-с.ССВВ	
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна варіантів використання*	Літера / Маса / Масштаб
Розробив		Косяк О.М.				
Перевірив		Проскура С.Л.				Аркуш 1 / Аркушів 1
Н. кон.		Халус О. А.			Клієнт-серверна аналітична система управління робочим часом	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*
Затвердив		Павлов О.А.				

ДПІС-5213.1181-с.ССП



					ДПІС-5213.1181-с.*ССП*	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна послідовності*	
Розробив		Косяк О.М.				Аркуш 1 / Аркушів 1
Перевірив		Проскура С.Л.				
Н. кон.		Халус О. А.			Клієнт-серверна аналітична система управління робочим часом	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*
Затвердив		Павлов О.А.				

ДПІС-5213.1181-с.СБД



					ДПІС-5213.1181-с.СБД		
Зм.	Арк.	№ документа	Підпис	Дата	*Схема бази даних*	Літера	Маса / Масштаб
Розробив		Косяк О.М.					
Перевірив		Проскура С.Л.				Аркуш 1	Аркушів 1
Н. кон.		Халус О. А.			Клієнт-серверна аналітична система управління робочим часом	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*	
Затвердив		Павлов О.А.					